



Electronics and communications department

Modern Hybrid Network Infrastructure

AI-Driven Security, Automation & Self-Deploying Networks

By

Marwa Saadallah Abdelmonem Alknany

Supervisor

DR / Ahmed Zakzouk

2025 / 2026

Dedication

We dedicate this graduation project to our parents, whose unconditional love, endless support, and sacrifices have been the foundation of our success. To the ones who believed in us when the road was long, and whose prayers were our guiding light through every challenge.

To our families and friends, who stood by us through the long nights of research and implementation, offering encouragement when it was needed most, and sharing every step of this journey with unwavering loyalty.

Finally, we dedicate this work to all students and engineers passionate about networking, cybersecurity, and the future of artificial intelligence.

Acknowledgments

First and foremost, we thank God for granting us the strength, patience, and knowledge to complete this project.

We would like to express our sincere gratitude to our academic supervisor, **Dr. Ahmed Zakzouk**, for his support and formal supervision .

We also extend our appreciation to the academic community and our peers. The foundational skills and motivation that culminated in this project were significantly shaped by competitive academic environments and challenges, such as our participation in the Egyptian Universities Cybersecurity Hackathon, which deeply inspired the advanced security and AI integration focus of our work.

Finally, we thank the faculty members of the Communications and Electronics Department at Al-Azhar University for providing us with the academic foundation necessary to tackle this complex integration of networking and cybersecurity.

Abstract

As enterprise networks grow in complexity, the need for robust infrastructure, proactive security monitoring, and automated threat response becomes paramount. This project presents the design, implementation, and simulation of a comprehensive enterprise network architecture integrated with Artificial Intelligence (AI) and automated security workflows. The environment is fully deployed and validated within the PNETLAB simulation platform.

The core infrastructure is built upon advanced routing using OSPF and secured inter-site communications via IPsec VPNs, protected by a pfSense firewall perimeter. To achieve proactive security monitoring, Wazuh SIEM is deployed to collect, analyze, and correlate security events across the network. Furthermore, network configuration and management are streamlined using Ansible, demonstrating the efficiency of Infrastructure as Code (IaC) in modern enterprise environments.

To address the critical need for rapid incident response, the project integrates Tines (SOAR) to intercept Wazuh alerts. These alerts are automatically forwarded to a large language model via the Groq AI API for intelligent threat analysis and summarization. The AI-generated incident reports are then instantly dispatched to security administrators through automated Telegram and Email notifications. The results of our simulation and security testing demonstrate a highly resilient, automated, and AI-aware network architecture capable of defending against and responding to modern cyber threats in real time.

Table of Contents

Dedication	2
Acknowledgments	3
Abstract	4
Table of Contents	5
List of Figures	7
List of Tables	8
List of Abbreviations	9
Chapter 1 — Introduction	10
1.1 Overview	10
1.2 Problem Statement	10
1.3 Project Objectives	11
1.4 Project Significance	12
1.5 Project Scope	13
1.6 Methodology	15
1.7 Report Structure	18
Chapter 2 — Literature Review & Background	20
2.1 Enterprise Network Architecture	20
2.2 Routing Protocols — OSPF	23
2.3 Network Security — IPsec VPN	27
2.4 Firewalls — pfSense	31
2.5 Security Information & Event Management (SIEM)	34
2.6 Network Automation — Ansible	39
2.7 Security Orchestration, Automation & Response (SOAR)	43
2.8 Artificial Intelligence in Cybersecurity	47
2.9 Threat Intelligence Platforms (TIP)	51
Chapter 3 — System Design	56
3.1 Design Overview	56
3.2 Network Topology	58
3.3 IP Addressing Scheme	59
3.4 Tools & Technologies Used	60
3.5 Simulation Environment — PNetLab	64
Chapter 4 — Implementation	66
4.1 Environment Setup	66
4.2 OSPF Configuration	67
4.3 IPsec Site-to-Site VPN	69

4.4 pfSense Firewall Configuration	72
4.5 Wazuh SIEM Deployment	75
4.6 Network Automation with Ansible.....	78
4.7 SOAR Configuration.....	83
4.8 Groq AI Integration.....	84
4.9 VirusTotal Threat Intelligence Integration	86
4.10 Alert Notification Setup (Telegram & Email)	87
Chapter 5 — Testing & Results.....	90
5.1 Testing Methodology	90
5.2 Network Connectivity Testing.....	90
5.3 VPN Tunnel Testing	92
5.4 Attack Simulation	92
5.5 Wazuh Detection Results	94
5.6 Automated Response Results	95
5.7 Results Analysis	97
Chapter 6 — Conclusion	99
6.1 Summary of Achievements	99
6.2 Challenges & Solutions	99
6.3 Recommendations	100
6.4 Future Work.....	100
References.....	102

List of Figures

Figure 3.1 — Proposed Network Topology

Figure 3.2 — IP Addressing Scheme Diagram

Figure 4.1 — Ansible Automation Workflow

Figure 4.2 — Wazuh + Groq AI Integration Flow

Figure 4.3 — VirusTotal API Response Example

Figure 5.1 — Wazuh Dashboard — Detected Alert

Figure 5.2 — AI-Generated Threat Report

List of Tables

- Table 3.1 — IP Addressing Scheme
- Table 4.1 — Tools and Technologies Used
- Table 5.1 — Attack Simulation Test Cases
- Table 5.2 — Detection & Response Time Results

List of Abbreviations

Abbreviation	Meaning
VPN	Virtual Private Network
OSPF	Open Shortest Path First
IPsec	Internet Protocol Security
SIEM	Security Information and Event Management
XDR	Extended Detection and Response
SOAR	Security Orchestration, Automation and Response
IaC	Infrastructure as Code
AI	Artificial Intelligence
LLM	Large Language Model
SOC	Security Operations Center
IDS/IPS	Intrusion Detection / Prevention System

Chapter 1 — Introduction

1.1 Overview

Modern enterprise networks serve as the critical digital backbone for today's smart cities and large-scale organizations, enabling seamless communication, data processing, and operational efficiency. However, as these infrastructures continue to scale in both size and complexity, they face unprecedented technical and security challenges. Traditional network management strategies, which often rely on manual device configuration and reactive security measures, are no longer capable of meeting the demands of high-availability environments.

Organizations are increasingly struggling with operational bottlenecks, human error in manual configurations, and an ever-evolving landscape of sophisticated cyber threats, where manual incident response times are often insufficient to mitigate risks before damage occurs.

To address these critical hurdles, there is a paradigm shift toward intelligent and automated network architectures. By integrating Infrastructure as Code (IaC) for consistent and scalable configurations, alongside AI-driven Security Orchestration, Automation, and Response (SOAR) workflows, enterprises can transition from reactive troubleshooting to proactive defense. This project explores the synergy between automated network management using Ansible and AI-enhanced threat detection through the integration of **Wazuh SIEM, Tines SOAR, and external Threat Intelligence via VirusTotal**. By automating repetitive administrative tasks and leveraging Large Language Models (LLMs) to perform intelligent, real-time threat analysis, the proposed solution aims to minimize human error, drastically reduce incident response latency, and ensure a resilient, self-optimizing network infrastructure.

1.2 Problem Statement

Despite the advancements in network management and cybersecurity, enterprise organizations continue to face significant operational and security challenges that undermine the resilience of their digital infrastructure. The core problems addressed by this project are categorized into three primary areas:

1. **Manual Configuration Errors and Operational Inefficiency:** Traditional enterprise networks rely heavily on manual command-line interface (CLI) configurations, which

are inherently prone to human error. These manual interventions lead to configuration drift, inconsistencies across distributed network devices, and a lack of scalability. Such errors often create security gaps that adversaries can exploit, transforming simple administrative tasks into critical vulnerabilities.

2. **Reactive Security Posture and Alert Fatigue:** Modern Security Information and Event Management (SIEM) systems, such as Wazuh, generate a massive volume of security alerts daily. Security analysts are frequently overwhelmed by this influx, leading to "alert fatigue," where critical security incidents are buried under a deluge of false positives or low-priority notifications. Consequently, most organizational responses remain reactive—addressing threats only after they have manifested, rather than preempting them.
3. **Data Privacy and Security Concerns in AI-Driven Analytics:** While integrating Large Language Models (LLMs) into security operations provides unparalleled threat analysis, it introduces significant data privacy concerns. Organizations are often hesitant to leverage external AI APIs due to the risk of exposing sensitive network logs, internal IP structures, or proprietary incident data to third-party providers. Balancing the need for AI-powered intelligence with the necessity of maintaining strict data sovereignty remains a primary obstacle for adopting AI in secure environments.

These issues highlight an urgent need for an integrated framework that automates infrastructure management, intelligentizes security response, and addresses the critical trade-off between AI efficiency and data privacy.

1.3 Project Objectives

The primary objectives of this project are outlined as follows:

- **Design and Simulate a Secure Multi-Site Enterprise Network:** Architect and deploy a robust, scalable enterprise network infrastructure within the PNETLAB simulation environment, ensuring high availability and secure inter-site communication using advanced routing and cryptographic tunneling.
- **Automate Deployment using Infrastructure as Code (IaC):** Eliminate human configuration errors and streamline network provisioning by utilizing Ansible

playbooks to automate the entire deployment, baseline management, and device configurations.

- **Build an AI-Assisted, Threat-Intelligence-Enriched SOC:** Construct a proactive Security Operations Center (SOC) capable of real-time event orchestration and automated incident triage. This involves integrating Wazuh SIEM for log ingestion, **leveraging external Threat Intelligence via the VirusTotal API for automated file and IP reputation assessment**, and utilizing Tines SOAR for workflow orchestration. The framework feeds this combined data into a Large Language Model (Groq AI) to generate intelligent, context-aware threat summaries, thereby minimizing alert fatigue and accelerating incident mitigation.

1.4 Project Significance

The significance of this project lies at the intersection of modern network infrastructure management and advanced cognitive cybersecurity operations. It provides clear value across three core dimensions: industry relevance, academic worth, and the demonstration of advanced practical engineering skills.

1. Industrial Relevance and Operational Impact

As modern enterprise environments transition toward multi-site architectures and smart city scales, traditional reactive security paradigms fail to keep pace with sophisticated threats. This project introduces immediate industrial value by:

- **Mitigating Alert Fatigue:** By filtering raw logs through a multi-stage funnel—Wazuh detection, automated VirusTotal reputation lookup, and Groq AI synthesis—the system minimizes false positives and ensures security analysts only deal with high-fidelity, actionable incidents.
- **Accelerating Incident Triage (Reducing MTTR):** Automating the extraction and lookup of indicators of compromise (IoCs) via the VirusTotal API cuts the Mean Time to Respond (MTTR) from hours of manual investigation to fractions of a second.
- **Ensuring Configuration Consistency:** Using Ansible for Infrastructure as Code (IaC) demonstrates how enterprise networks can completely eliminate manual

configuration drift and human errors, which are major entry points for cyber adversaries.

2. Academic and Research Value

From an academic perspective, this research explores the architectural synergy between three distinct, cutting-edge technical paradigms: Network Automation (Ansible), Security Orchestration (Tines SOAR), and Hybrid Cyber Intelligence (Deterministic API metadata combined with Probabilistic LLM analysis).

- Unlike typical academic security frameworks that rely solely on isolated SIEM alerts, this project designs and documents a **"Hybrid Intelligence" framework**. It provides a reproducible blueprint showing how static, signature-based threat intelligence (VirusTotal) can be harmonized with semantic reasoning engines (Large Language Models) to deliver fully orchestrated security workflows.

3. Advanced Practical Skills Demonstrated

This project serves as a comprehensive validation of end-to-end network engineering and cybersecurity capabilities, proving mastery over complex, production-grade technologies, including:

- **Enterprise Networking & Simulation:** Designing and validating robust, multi-site topologies inside PNETLAB using dynamic routing protocols (OSPF) and secure cryptographic tunnels (IPSec VPN).
- **Infrastructure Automation:** Developing modular, reusable Ansible playbooks for automated network provisioning and security baseline enforcement.
- **SOC Architecture & DevSecOps:** Setting up and tuning a functional Security Operations Center (SOC) framework utilizing Wazuh, writing customized Tines SOAR storyboards, and integrating heterogeneous third-party REST APIs (VirusTotal v3 and Groq AI).

1.5 Project Scope

This project is scoped as a fully simulated proof-of-concept rather than a production deployment, with clearly defined boundaries outlining what is included within the core implementation and what falls outside the project's coverage.

1. In-Scope Components

The active development, deployment, and validation of this framework encompass the following areas:

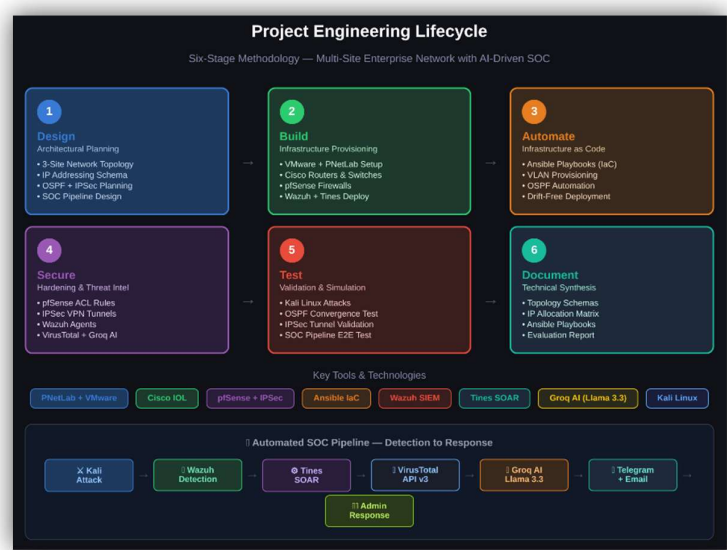
- **Multi-Site Network Architecture & Simulation:** Designing and implementing a three-site enterprise network topology (Headquarters, Branch A, and Branch B) simulated entirely within the **PNetLab** environment, hosted on a **VMware Workstation** virtual machine. The infrastructure utilizes **Cisco IOL** routers and switches, with **pfSense** firewalls providing perimeter security, dynamic **OSPF** routing, and secure Site-to-Site **IPsec VPN** tunnels.
- **Infrastructure as Code (IaC) Automation:** End-to-end automation of all network provisioning, device configuration management, and baseline security enforcement using **Ansible playbooks**.
- **Centralized Monitoring & Threat Intelligence Enrichment:** Implementing real-time security monitoring using a **Wazuh SIEM/XDR** cluster for log collection and alert generation, seamlessly enriched with external reputation-based threat intelligence via the **VirusTotal API v3**.
- **SOAR Orchestration & AI Analysis:** Designing automated incident response workflows using **Tines SOAR** to ingest alerts. These workflows pass contextual data into the **Groq AI API (Llama3 model)** to generate intelligent, human-readable threat summaries and mitigation recommendations.
- **Alert Delivery & Visualization:** Routing synthesized security insights directly to security analysts via **Telegram notifications** and populating custom visualizations on the **Wazuh dashboard**.
- **Validation & Attack Simulation:** Conducting controlled security testing—specifically common reconnaissance and brute-force attack techniques—to validate the system's detection, enrichment, and automated reporting capabilities.

2. Out-of-Scope Components

To maintain project feasibility and optimize available resources, the following elements are explicitly excluded from the current scope:

- **Physical Hardware Deployment:** The project is strictly virtualized; it does not involve deployment on physical production switches, routers, or hardware appliances.
- **Live ISP Integration:** All wide-area network (WAN) connectivity is simulated internally within the PNetLab environment without utilizing real Internet Service Providers (ISPs) or public IP routing.
- **Advanced Inline Defenses & Controller Architectures:** The scope excludes the deployment of Software-Defined Networking (SDN) controllers and advanced standalone Intrusion Prevention Systems (such as Suricata).
- **Air-Gapped Local AI Hosting:** The framework utilizes cloud-hosted models via the Groq API rather than hosting heavy open-source LLM weights on a fully local, air-gapped infrastructure.
- **Production-Grade Testing & Auditing:** High-load traffic performance testing, formal penetration testing against live enterprise environments, and formal compliance auditing (e.g., ISO 27001, NIST) are explicitly excluded.

1.6 Methodology



This project adopts a structured, six-stage engineering lifecycle to methodically transition from theoretical system design to an automated, intelligent, and securely validated proof-of-

concept. This systematic approach ensures that every infrastructure and security component is planned, deployed, and validated sequentially.

The project engineering lifecycle proceeds through the following sequential phases:

1. Design (Architectural Planning)

The lifecycle begins with defining the high-level network and security architecture. This phase involves:

- Mapping out the three-site enterprise topology comprising the Headquarters, Branch A, and Branch B.
- Planning the global IP addressing schema and subnetting boundaries across all distributed sites.
- Selecting Open Shortest Path First (OSPF) as the dynamic routing protocol and IPsec VPN for secure inter-site transport.
- Modeling the integrated SOC pipeline topology linking Wazuh SIEM, Tines SOAR, VirusTotal API v3, and the Groq AI analytical engine.

2. Build (Infrastructure Provisioning)

Once the architectural designs are finalized, the virtual simulation environment is constructed. This phase entails:

- Setting up VMware Workstation to host the unified PNetLab simulation platform.
- Instantiating the underlying network nodes, including virtualized Cisco IOL routers and switches within the hypervisor.
- Deploying pfSense firewall appliances at each site's perimeter to control traffic flow and mark network boundaries.
- Provisioning the core security management components, including the centralized Wazuh SIEM manager and the Tines SOAR automation workspace.

3. Automate (Infrastructure as Code)

To eliminate human configuration errors and achieve high operational efficiency, the entire network layer configuration is automated:

- Adopting the Infrastructure as Code (IaC) paradigm by writing modular, reusable Ansible playbooks.
- Developing automation tasks to provision VLANs, configure network device interfaces, and establish global dynamic OSPF routing metrics across the Cisco infrastructure.
- Utilizing Ansible to push baseline configurations rapidly, ensuring a deterministic and drift-free deployment across all three sites.

4. Secure (Hardening & Threat Intelligence Integration)

Security baselines are systematically injected and verified across the automated multi-site environment:

- Configuring rigid packet filtering and perimeter firewall access control rules on the pfSense appliances.
- Orchestrating secure cryptographic paths via Site-to-Site IPsec VPN tunnels to handle protected inter-site communication over the simulated WAN.
- Deploying and tuning endpoint monitoring agents (Wazuh agents) across critical workloads to handle continuous log ingestion.
- Integrating third-party APIs into the automated incident response workflows, specifically targeting the deterministic VirusTotal API v3 for reputation enrichment and the Groq AI cloud API for cognitive threat evaluation.

5. Test (Validation & Attack Simulation)

Rigorous testing is performed to validate the baseline health of the network and the absolute accuracy of the automated SOC pipeline:

- Verifying network convergence and connectivity across the OSPF paths and testing the fallback stability of the IPsec tunnels.
- Executing controlled attack simulations using Kali Linux, focusing on network reconnaissance and brute-force attack vectors.
- Evaluating the end-to-end detection-to-notification funnel, confirming that Wazuh ingests the raw logs, Tines successfully fetches reputation data from the VirusTotal

API, Groq AI generates context-aware mitigation summaries using the Llama3 model, and Telegram dispatches the final high-fidelity alert to the administrator.

6. Document (Technical Synthesis)

The final phase focuses on consolidating all engineering artifacts generated during the lifecycle:

- Compiling the precise architectural topology schemas, IP allocation matrices, and pfSense security policy sheets.
- Organizing and archiving the complete catalog of Ansible playbooks and Tines SOAR workflow storyboards.
- Formatting and synthesizing empirical evaluation data, dashboard graphs, and event output logs into a comprehensive technical report.

1.7 Report Structure

The remainder of this technical report is systematically organized into five distinct chapters, tracing the complete engineering lifecycle of the proposed automated and cognitive SOC framework. Each chapter addresses a specific phase of development, evaluation, and synthesis as outlined below:

- **Chapter 2: Literature Review & Background** – Establishes the theoretical foundations of the project, exploring multi-site enterprise architectures, dynamic routing via OSPF, IPsec cryptographic tunneling, Infrastructure as Code (IaC) with Ansible, centralized monitoring via Wazuh SIEM, and modern SOAR architectures utilizing external Threat Intelligence (VirusTotal) and cloud-hosted Large Language Models (Groq AI).
- **Chapter 3: System Design** – Details the structural design of the simulated three-site network within PNetLab, including specific router/switch placement, perimeter firewall zoning (pfSense), logical asset mapping, and the architectural taxonomy of the automated incident response pipeline.
- **Chapter 4: Implementation** – Document the technical implementation and configuration steps. This chapter covers the deployment of pfSense firewalls, development of modular Ansible playbooks, configuration of Wazuh log-ingestion

rules, and the continuous integration of Tines SOAR storyboards with external API endpoints.

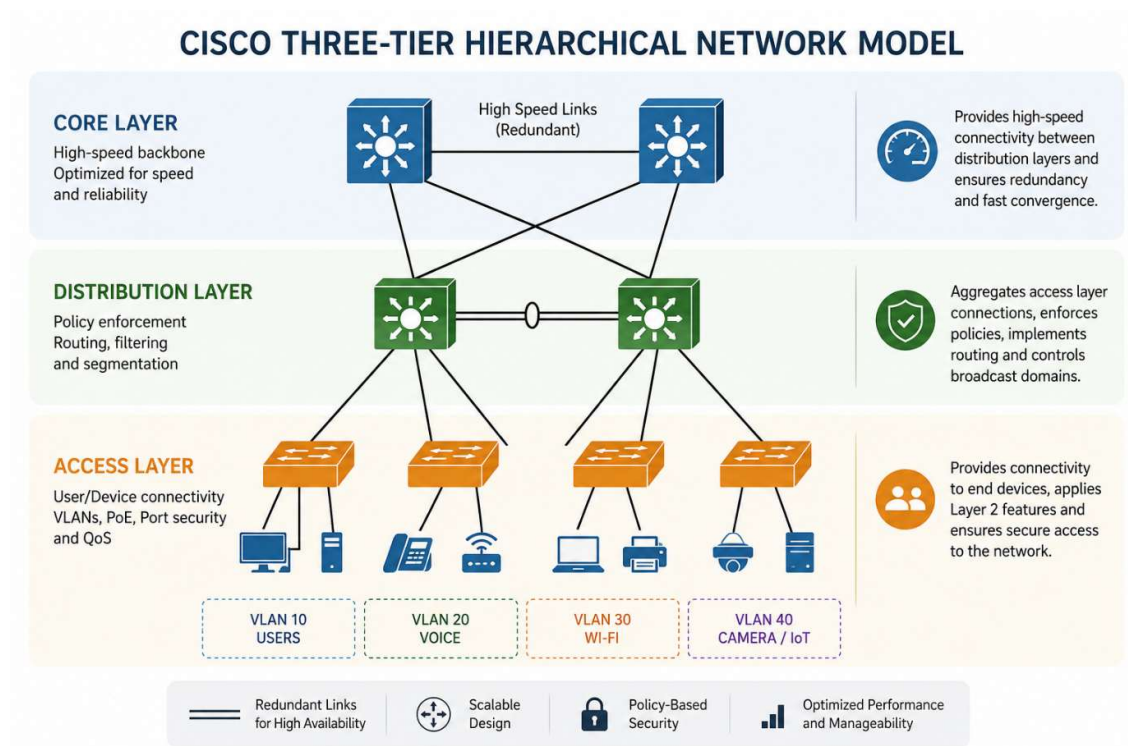
- **Chapter 5: Testing & Results** – Provides an empirical and data-driven evaluation of the framework. It measures network convergence efficiency, cryptographic stability under load, and analyzes the end-to-end detection-to-notification latency during live, simulated attack scenarios (reconnaissance and brute-force attempts).
- **Chapter 6: Conclusion & Future Work** – Summarizes the critical milestones achieved by the project, reviews engineering constraints faced during implementation, offers practical troubleshooting recommendations, and presents a comprehensive roadmap for future enhancements.

Chapter 2 — Literature Review & Background

2.1 Enterprise Network Architecture

Modern enterprise network architectures are engineered to deliver resilient, scalable, high-throughput, and secure connectivity across geographically distributed corporate environments. Unlike localized campus topologies restricted to a single physical facility, an enterprise-grade infrastructure must seamlessly unify distributed locations over wide-area transit networks. To manage the inherent technical complexity, minimize latency, and ensure strict security boundaries, contemporary enterprise deployments discard flat architectural designs in favor of structured frameworks governed by hierarchical design models, hub-and-spoke topologies, and multi-site enterprise patterns.

2.1.1 Hierarchical Design Models



To eliminate broadcast overhead, isolate infrastructure faults, and establish predictable traffic paths, enterprise networks utilize a structured three-tier or two-tier (collapsed core) hierarchical design model. This model systematically partitions the network into three distinct functional layers: Core, Distribution, and Access.

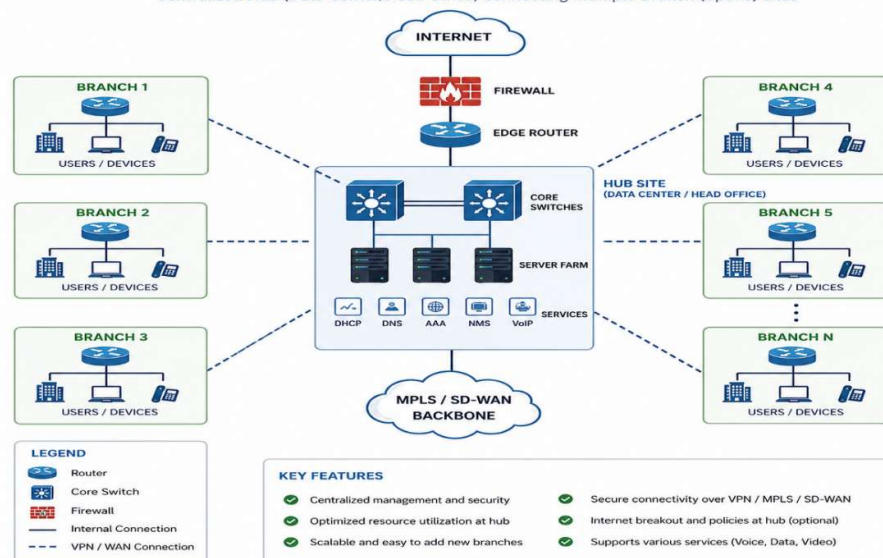
2.1.2 Hub-and-Spoke Topology

When enterprise organizations expand across distinct geographical regions, managing wide-area transport demands an efficient and highly controllable logical layout. The **Hub-and-Spoke topology** represents the industry-standard architectural blueprint for multi-site WAN transport, characterized by a centralized hub connecting to multiple independent remote spoke locations.

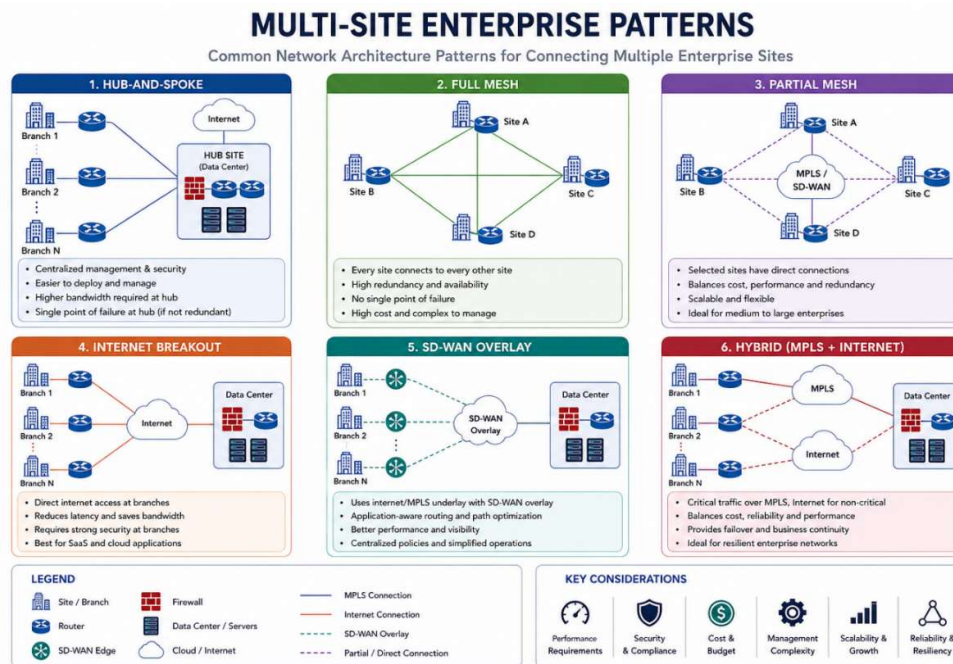
- **The Centralized Hub:** Typically instantiated at the corporate Headquarters (HQ) or a central data center infrastructure. The hub acts as the main transit node and the hosting environment for critical centralized assets, including Security Operations Centers (SOCs), centralized SIEM managers, database clusters, and core authentication services.
- **The Distributed Spokes:** Represent remote facilities or branch offices. Spokes maintain localized switching layers but rely entirely on a single or redundant upstream path to communicate with the central hub.
- **Traffic Traversal & Control:** In a classic hub-and-spoke pattern, all inter-site (Spoke-to-Spoke) traffic is systematically forced to traverse the central hub. This design architecture eliminates the cost and complexity of maintaining a full physical mesh, while providing a centralized vantage point where security teams can continuously inspect, audit, and log cross-site traffic.

GENERIC HUB AND SPOKE NETWORK TOPOLOGY DESIGN

Centralized Hub (Data Center/Head Office) connecting multiple Branch (Spoke) sites



2.1.3 Multi-Site Enterprise Patterns



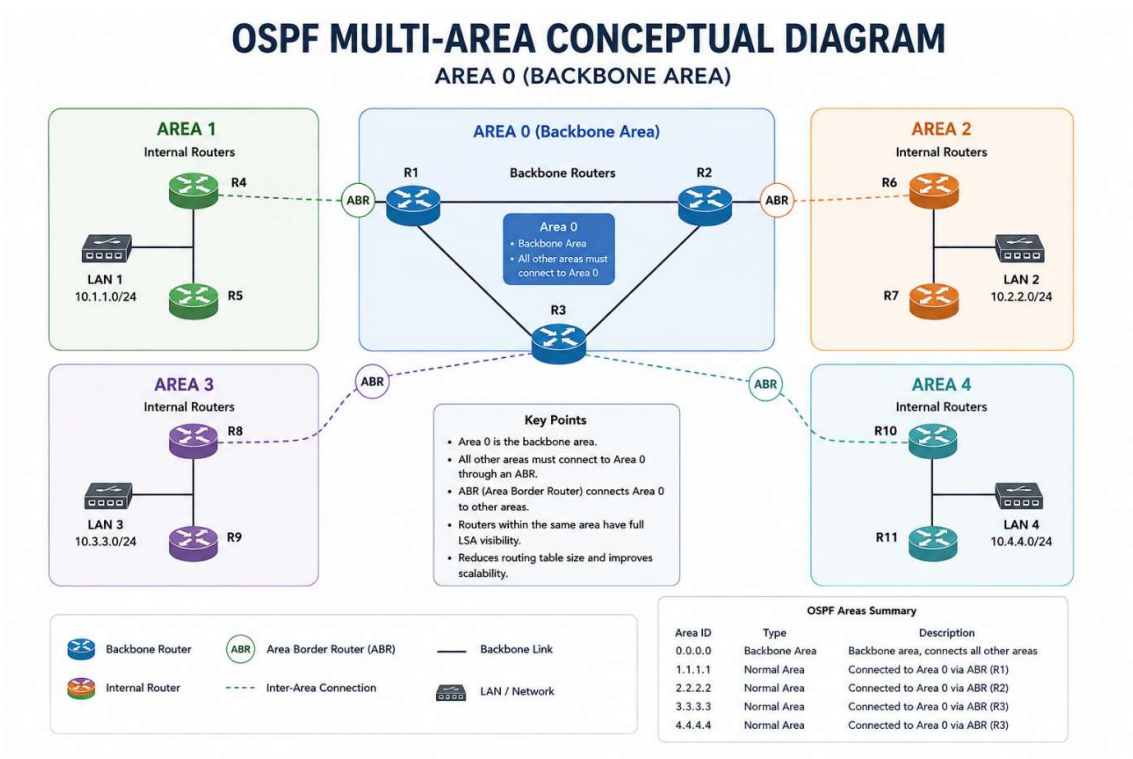
Integrating hierarchical switching principles within a wide-area hub-and-spoke infrastructure yields a comprehensive **Multi-Site Enterprise Pattern**. This pattern provides a reproducible engineering framework that dictates how data planes, control planes, and management planes behave when exposed to external transit fabrics.

Because leasing dedicated private fiber connections across disparate geographic regions is cost-prohibitive, multi-site enterprise patterns leverage an intermediate, untrusted transit provider—such as a simulated or public Internet Service Provider (ISP)—as the WAN transport medium. To preserve corporate confidentiality and integrity across this public fabric, the multi-site pattern implements strict **Edge Demarcation Lines**. Every individual site provisions perimeter stateful security appliances directly underneath its upstream WAN routing layer. These gateways establish encrypted cryptographic paths (such as Site-to-Site IPsec VPN tunnels) to securely encapsulate private corporate traffic before it enters the public routing space. By combining dynamic interior gateway protocols like OSPF for path convergence with perimeter stateful access controls, multi-site patterns ensure that the internal enterprise directory remains highly available, resilient to link failures, and completely insulated from external threats.

2.2 Routing Protocols — OSPF

To facilitate dynamic, scalable, and resilient communication across distributed enterprise network architectures, routing infrastructures must implement efficient interior gateway routing protocols. Static routing becomes operationally unsustainable as network nodes multiply, as it lacks the programmatic capacity to automatically adapt to dynamic link fluctuations or link failures. Open Shortest Path First (OSPF) is a link-state routing protocol engineered as an open standard by the Internet Engineering Task Force (IETF) under RFC 2328. Operating directly over the IP network layer (IP protocol 89), OSPF utilizes Dijkstra's Shortest Path First (SPF) algorithm to construct a loop-free, mathematically optimized routing topology matrix based on calculated link states.

2.2.1 OSPF Hierarchical Architecture and Areas



One of OSPF's primary architectural advantages is its hierarchical area design, which limits the size of the Link-State Database (LSDB), minimizes routing protocol overhead, and isolates topology changes. Rather than forcing every router to maintain the state of every

single link across the global corporate directory, OSPF segments the Autonomous System (AS) into distinct logical zones called **Areas**.

- **Backbone Area (Area 0):** The foundational core transit zone of any OSPF network. All non-backbone areas must physically or logically connect directly to Area 0. All inter-area routing traffic must traverse the backbone, ensuring a loop-free topology across the entire enterprise distribution.
- **Non-Backbone Areas (Normal Areas):** Localized groupings of routers and subnets. Topology changes within a normal area (such as an interface flapping) are confined locally and do not trigger computationally heavy SPF recalculations on routers situated in other areas.
- **Area Border Routers (ABRs):** Specialized nodes that maintain separate LSDBs for each area they connect to (e.g., Area 0 and Area 1). ABRs summarize local topological paths before passing them into the backbone transit zone.
- **Autonomous System Boundary Routers (ASBRs):** Routers that exchange routing information with networks outside the OSPF domain, such as external routes learned via static routing, BGP, or other redistribution mechanisms.

2.2.2 Link-State Advertisements (LSAs)

OSPF routers exchange network infrastructure intelligence through data packets known as Link-State Advertisements (LSAs). LSAs are flooded throughout the network, allowing every router within a specific area to assemble an identical, synchronized map of the network topology (the LSDB). To maintain hierarchical boundaries, OSPF utilizes several distinct LSA types:

- **Type 1 (Router LSA):** Generated by every individual OSPF router. It describes the state and cost of the router's local interfaces and is flooded strictly within its local area boundaries.
- **Type 2 (Network LSA):** Generated by the Designated Router (DR) on multi-access networks (such as Ethernet segments). It lists all routers attached to that specific multi-access network segment.
- **Type 3 (Summary LSA):** Generated by ABRs to advertise internal prefixes from one area into another, enabling inter-area communication without flooding full topological details.

- **Type 4 (ASBR Summary LSA):** Generated by ABRs to advertise the presence and metric cost of an ASBR to the rest of the OSPF autonomous system.
- **Type 5 (AS External LSA):** Generated by ASBRs to advertise external routes redistributed into the OSPF domain from non-OSPF sources. These are flooded throughout the entire OSPF domain.

2.2.3 OSPF Metric Calculation

OSPF determines the most efficient routing path across an enterprise topology based on a cost metric assigned to each egress interface along a given network path. The cost is inversely proportional to the bandwidth of the physical or virtual interface, calculated using the standard formula:

$$COST = \frac{Reference\ Bandwidth}{Interface\ Bandwidth}$$

By default, the standard architectural reference bandwidth defined in standard OSPF implementations is 100 Mbps (10^8 bits per second). This yields a default cost value of 1 for Fast Ethernet interfaces. However, modern high-speed enterprise interfaces (such as Gigabit Ethernet or 10-Gigabit Ethernet) exceed this default reference value, causing OSPF to assign an identical baseline cost of 1 to all links operating above 100 Mbps.

To prevent suboptimal path selection in modern infrastructures, network engineers modify the OSPF **Auto-Cost Reference Bandwidth** globally. Adjusting this parameter to a modern standard—such as 10 Gbps (10^{10} bits per second)—ensures that the SPF algorithm computes distinct, deterministic cost metrics that accurately differentiate between lower-speed WAN links and high-speed local transit boundaries.

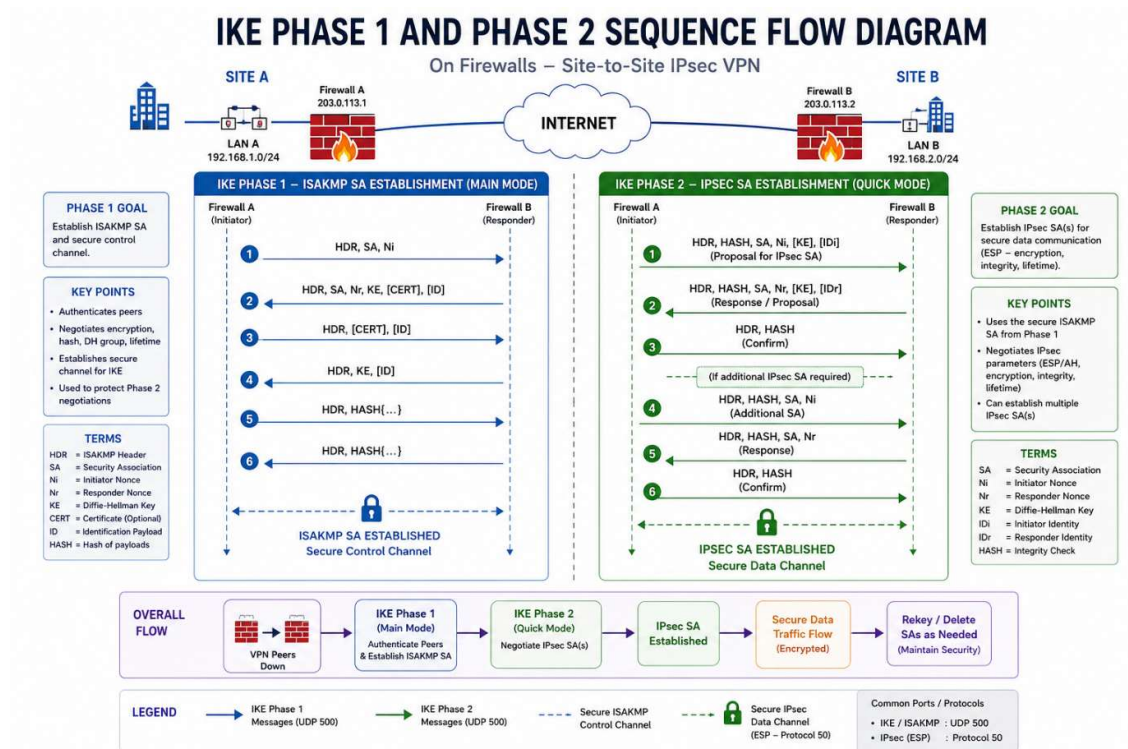
2.2.4 Comparative Analysis: OSPF vs. EIGRP vs. BGP

Selecting the appropriate routing protocol requires evaluating convergence characteristics, scalability limits, and vendor interoperability. The table below outlines a comparative analysis between OSPF and other industry-standard routing protocols:

Feature / Criteria	OSPF	EIGRP	BGP
Protocol Type	Link-State (IGP)	Advanced Distance-Vector / Hybrid (IGP)	Path-Vector (EGP)
Underlying Algorithm	Dijkstra Shortest Path First (SPF)	Diffusing Update Algorithm (DUAL)	Best Path Selection Algorithm
Metric Attributes	Cost (Inversely proportional to bandwidth)	Composite (Bandwidth, Delay, Load, Reliability)	Path Attributes (AS-Path, Local Preference, MED)
Convergence Speed	Fast (Triggers instant LSAs upon link state modification)	Extremely Fast (Maintains pre-calculated feasible successors)	Slow (Prioritizes global path stability over raw speed)
Interoperability	Open Standard (Universal multi-vendor deployment)	Cisco Proprietary (Open standard under RFC 7868 but lacks universal vendor support)	Open Standard (The foundational global standard for Internet routing)
Primary Use Case	Interior core and distribution layer routing in multi-vendor enterprise networks.	Localized interior gateway routing within pure or legacy Cisco-centric infrastructures.	Edge routing bridging external autonomous systems and Internet service providers.

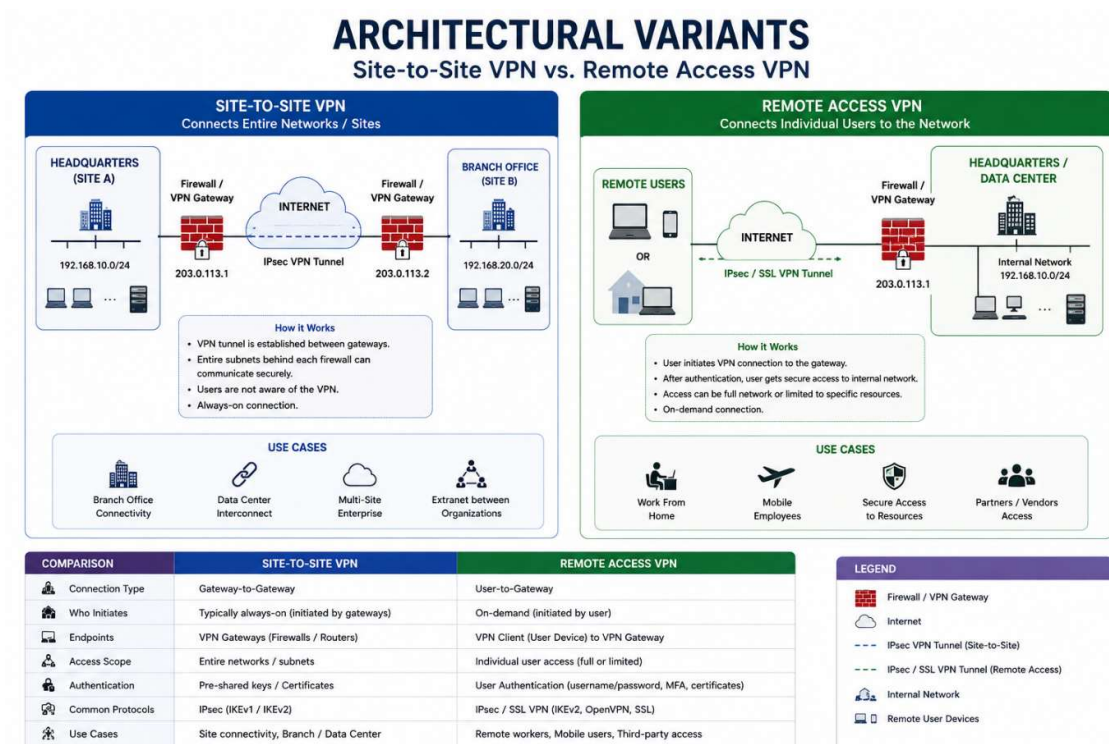
While EIGRP offers fast convergence through backup path tracking, its historically vendor-dependent design restricts its utility in modern heterogeneous networks. Border Gateway Protocol (BGP) is built for massive scalability and policy enforcement across external administrative boundaries, but its slow convergence metrics render it unsuited for internal campus topology convergence. Therefore, OSPF remains the industry-standard choice for internal enterprise infrastructures, providing rapid fault recovery, multi-vendor flexibility, and a structured area architecture capable of handling high-density routing tables without degrading network stability.

2.3 Network Security — IPsec VPN



As enterprise organizations transmit proprietary data across public or shared wide-area transit architectures—such as routed paths handled by internet service providers—the implementation of robust layer-3 cryptographic mechanisms becomes imperative. Internet Protocol Security (IPsec) is a modular framework of open standards developed by the Internet Engineering Task Force (IETF) designed to secure communications over IPv4/IPv6 networks. IPsec ensures data confidentiality through encryption, data integrity by detecting unauthorized alterations, and endpoint authentication to verify the identity of communicating cryptographic peers.

2.3.1 Architectural Variants: Site-to-Site vs. Remote Access VPN



Enterprise deployments leverage virtual private networks in two primary architectural patterns depending on whether they are interconnecting fixed facilities or mobile end-users:

- **Site-to-Site VPN:** This architecture connects entire fixed network infrastructures across distinct geographical locations over an untrusted public transit medium. Dedicated perimeter gateways (such as firewalls or routers) terminate the cryptographic tunnel on behalf of the local network. Endpoints inside the local area network (LAN) send and receive traffic transparently without running proprietary VPN client software, as the edge gateways handle all encapsulation, encryption, and decryption. This pattern is highly optimized for corporate topologies connecting distributed branch offices to a centralized headquarters.

- **Remote Access VPN:** This architecture connects individual, mobile end-users to the corporate network from variable, untrusted external locations (e.g., remote workers). It operates on a client-server model, requiring the endpoint device to install and run a dedicated VPN client software application or utilize a web browser (SSL/TLS VPN). The client establishes an individual secure tunnel directly back to a centralized corporate VPN concentrator to gain network access, making it highly dynamic but increasing endpoint management overhead.

2.3.2 Internet Key Exchange (IKE) Operational Phases

The dynamic negotiation, establishment, and maintenance of cryptographic associations between perimeter edge nodes are orchestrated via the **Internet Key Exchange (IKE)** protocol. Governed primarily by IKEv2 (RFC 7296) in modern architectures, this cryptographic handshake functions in two distinct, sequential phases:

IKE Phase 1 (Establishment of the Management Channel)

The objective of Phase 1 is to authenticate the two gateway peers and set up a secure, encrypted channel to protect subsequent routing negotiations.

- **Negotiation:** The peers exchange proposals to agree on a common set of security parameters: a symmetric encryption algorithm, a hashing/integrity algorithm, an authentication method (Pre-Shared Keys or Digital Certificates), and a Diffie-Hellman (DH) group.
- **Diffie-Hellman Key Exchange:** The peers programmatically generate a shared secret key over an untrusted channel using public-key cryptography. The DH group number dictates the mathematical strength and bit-length of the key.
- **Security Association (SA):** Once authenticated, a bi-directional management tunnel known as the **IKE Phase 1 SA** (or ISAKMP SA) is established, securing all future control-plane messages between the gateways.

IKE Phase 2 (Establishment of Data Tunnels)

Utilizing the secure control channel established during Phase 1, the edge peers execute Phase 2 negotiations to build the operational data tunnels.

- **Parameter Agreement:** The peers negotiate the specific encapsulation parameters for production traffic. This includes selecting the security protocol (almost universally Encapsulating Security Payload - ESP), choosing the data encryption and hashing algorithms, and specifying the tunnel encapsulation mode (Tunnel Mode vs. Transport Mode).
- **Perfect Forward Secrecy (PFS):** If PFS is enabled, the gateways execute an additional, independent Diffie-Hellman key exchange during Phase 2. This guarantees that the encryption keys used for production data are completely independent of the Phase 1 management keys. If a single session key is compromised, previous and future sessions remain cryptographically secure.
- **IPsec SAs:** This phase concludes with the creation of two distinct **uni-directional IPsec SAs**—one for inbound traffic and one for outbound traffic—enabling encrypted data traversal between corporate facilities.

2.3.3 Cryptographic and Encryption Algorithms

The strength of an IPsec VPN tunnel depends entirely on the mathematical robustness of the cryptographic algorithms bound to its Security Associations. Modern enterprise security frameworks discard legacy, vulnerable algorithms like Data Encryption Standard (DES), Triple DES (3DES), and Message Digest 5 (MD5) due to their susceptibility to brute-force and collision attacks, mandating instead advanced cryptographic standards:

- **Advanced Encryption Standard (AES):** AES is a symmetric block cipher selected by the National Institute of Standards and Technology (NIST) as the global standard for data encryption. It operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, and 256 bits (with AES-256 being the standard recommendation for enterprise and government infrastructures).
- **Cipher Block Chaining (AES-CBC) vs. Galois/Counter Mode (AES-GCM):** Traditional implementations deploy **AES-CBC**, which requires a separate hashing algorithm (like SHA-256) to provide message authentication and integrity verification. Modern high-throughput networks favor **AES-GCM**, an **Authenticated Encryption with Associated Data (AEAD)** algorithm. AES-GCM performs encryption and integrity verification simultaneously within the same mathematical operation. This eliminates the need for a separate hashing pass, reducing CPU

utilization on security gateways and enabling hardware-accelerated cryptographic throughput.

- **Secure Hash Algorithm 2 (SHA-2):** For deployments requiring independent integrity checks, SHA-2 (specifically SHA-256, SHA-384, or SHA-512) is utilized to compute cryptographic checksums, ensuring that packets are not intercepted, altered, or replayed by unauthorized third parties during WAN traversal.

2.4 Firewalls — pfSense

A perimeter firewall represents an organization's primary line of defense at the network edge, enforcing mandatory access control policies at the administrative boundary between trusted internal local area networks and untrusted external environments. As threats evolve, firewall technology has transitioned from primitive packet-filtering mechanisms to highly intelligent state-tracking engines capable of inspecting traffic context up to the application layer.

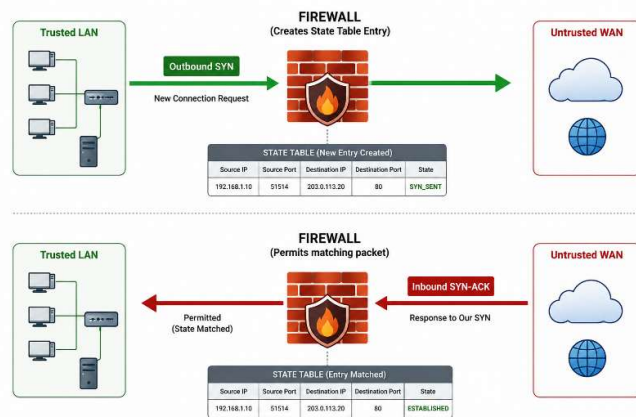
2.4.1 Stateful Packet Inspection (SPI) Concepts

Traditional legacy firewalls operated exclusively as stateless packet filters. They evaluated individual network packets in absolute isolation based solely on static layer-3 and layer-4 headers (such as source/destination IP addresses and port numbers). This paradigm is highly inefficient and insecure, as it cannot verify whether an incoming packet is a legitimate response to an internal request or an unsolicited external probe, requiring administrators to open broad, permanent holes in the security perimeter.

Modern network defense relies on **Stateful Packet Inspection (SPI)**. Stateful firewalls continuously monitor and track the complete lifecycle and operational state of active network connections traversing their interfaces. The firewall constructs and maintains a dynamic **State Table** (session table) in memory. When a connection is initiated—for instance, an internal host establishing a TCP session via the standard 3-way handshake (SYN, SYN-ACK, ACK)—the SPI engine records key connection parameters, including:

- Dynamic source and destination IP address pairings.
- Explicit source and destination Layer-4 port configurations.

- TCP sequence numbers and protocol session flags.



By cross-referencing every traversing packet against this active state table, the firewall automatically permits legitimate return traffic matching an established, validated session without requiring manual rule configurations. Unsolicited inbound traffic that fails to match an existing state table entry is instantaneously dropped, enforcing a strict "default-deny" security posture at the network boundary.

2.4.2 pfSense Architecture

In virtualized, software-defined, and enterprise network deployment models, **pfSense** has established itself as an enterprise-grade, open-source stateful firewall and routing platform. The architectural robustness of pfSense stems from its underlying operating system, **FreeBSD**, which provides a highly stable, secure, and optimized network stack.

At the core of the pfSense packet processing engine is the **Packet Filter (pf)** subsystem, originally developed for OpenBSD. The pf engine executes stateful packet inspection, advanced Network Address Translation (NAT), and traffic normalization directly within the FreeBSD kernel space to minimize latency and maximize throughput.

Architecturally, pfSense operates on a modular framework. Beyond core firewalling and state tracking, it aggregates multiple network-critical microservices onto a single logical control plane, including:

- Dynamic routing protocol convergence (via FRRouting for OSPF/BGP).
- High-availability failover provisioning (via Common Address Redundancy Protocol - CARP).

- Multi-realm Virtual Private Network (VPN) termination (supporting IPsec, OpenVPN, and WireGuard).
- Granular local infrastructure services (DHCP server allocation and DNS resolver capabilities).

Because pfSense runs on commodity x86-64 hardware architectures or virtualized hypervisors, its processing performance scales dynamically with host CPU and RAM resources, making it an agile choice for cloud, hybrid, and simulated multi-site network security stacks.

2.4.3 Comparative Analysis: pfSense vs. Cisco ASA vs. FortiGate

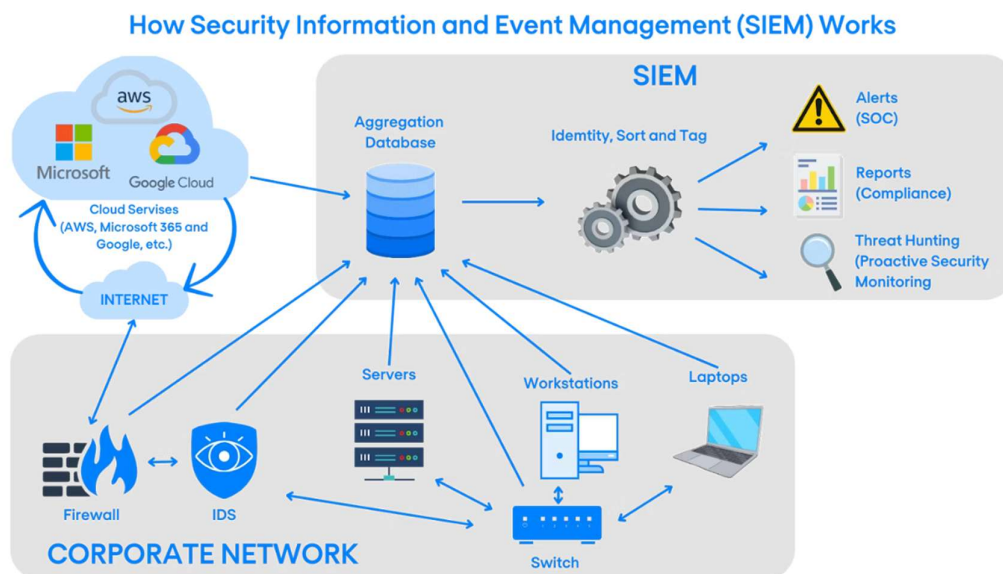
To justify the selection of an edge security platform within an enterprise topology, network architects must evaluate licensing models, underlying hardware execution layers, and threat intelligence capabilities. The table below outlines a comparative analysis between pfSense, Cisco ASA, and FortiGate:

Evaluation Criteria	pfSense	Cisco ASA (Adaptive Security Appliance)	FortiGate (Next-Generation Firewall)
Licensing & Business Model	Open-Source (Community Edition is free; commercial support available via Netgate).	Proprietary (Requires commercial licensing for software features, throughput, and AnyConnect VPN).	Proprietary (Requires active commercial subscriptions for hardware and FortiGuard security services).
Underlying Operating System	FreeBSD Kernel with pf packet filtering subsystem.	Cisco ASA Software (Linux-based kernel in newer virtualized/FTD models).	FortiOS (Highly customized, hardened real-time operating system).
Hardware Architecture	Commodity Hardware-Dependent (Processes traffic via standard x86-64 host CPUs).	Dedicated Appliance / Software-Driven (Utilizes generic x86 CPUs or specialized x86 system-on-chip).	Hardware-Accelerated (Utilizes proprietary, dedicated ASICs - Network Processors [NP] and Content Processors [CP]).
Deep Packet Inspection (DPI) & Threat Intel	Package-Dependent (Requires manual integration of open-source	Centralized Threat Defense (Cisco Firepower integration backed by Cisco Talos intelligence).	Native Next-Gen Capabilities (Built-in SSL inspection, sandboxing, and

	utilities like Snort, Suricata, or Squid).		AV backed by FortiGuard Labs).
Primary Deployment Use Case	Highly agile, cost-effective parameter defense for SMBs, virtual labs, and software-defined branch infrastructures.	Traditional enterprise data center security boundaries requiring highly stable, legacy-compliant stateful inspection.	High-throughput, enterprise-wide Next-Generation security fabrics requiring wire-speed hardware-accelerated deep packet inspection.

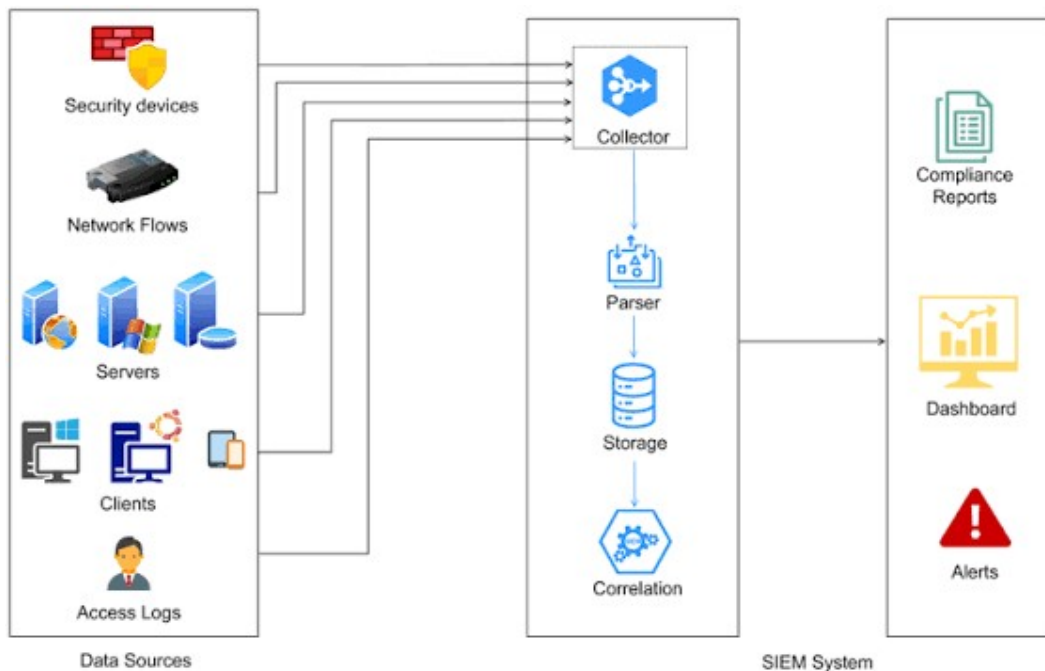
While Cisco ASA and FortiGate offer superior throughput in ultra-high-density data centers due to specialized hardware acceleration (such as FortiGate's proprietary ASICs), they carry substantial recurring capital and operational licensing expenditures. On the other hand, pfSense leverages the robust FreeBSD kernel to provide equivalent stateful security, granular access control, and robust VPN termination capabilities over standard compute resources without software licensing bottlenecks, establishing a highly cost-efficient and mathematically sound edge defense strategy for modern multi-site networks.

2.5 Security Information & Event Management (SIEM)



In modern distributed corporate network environments, every structural component—including edge firewalls, core switches, enterprise endpoints, and operating systems—continuously generates verbose operational logs and telemetry. Managing, auditing, and inspecting these millions of disparate events manually is operationally impossible for corporate security teams. Security Information and Event Management (SIEM) frameworks were engineered to solve this limitation by providing a centralized platform for real-time security monitoring, automated log ingestion, event correlation, and programmatic threat detection.

2.5.1 SIEM Core Concepts and Log Correlation Engine



A foundational SIEM architecture operates across a four-stage data pipeline designed to transform raw machine telemetry into actionable security intelligence:

1. **Data Collection and Ingestion:** The SIEM platform ingests heterogeneous, unstructured log streams from thousands of distributed physical and virtual network

infrastructure assets using transport mechanisms like Syslog, WinLogBeat, or secure endpoint daemons.

2. **Data Parsing and Normalization:** Because a Cisco switch, a pfSense firewall, and a Linux server format system timestamps and event logs differently, the SIEM's ingestion parser normalizes the raw inputs into a standardized schema (e.g., mapping variable fields into uniform attributes like `source_ip`, `destination_ip`, `event_id`, and `severity_level`).
3. **Data Indexing and Storage:** The normalized logs are chronologically indexed and written to high-performance database clusters to facilitate rapid forensic searching and compliance auditing.
4. **Log Correlation Execution:** This is the analytical core of the SIEM platform. The correlation engine runs the unified data stream against a centralized matrix of deterministic and probabilistic rule tracks. **Log Correlation** involves combining and cross-referencing multiple seemingly unrelated events across separate infrastructure boundaries to identify hidden security breaches. For instance, an isolated brute-force authentication failure on an edge gateway may be flagged as low-risk; however, if the correlation engine detects that the identical external IP address successfully authenticated via SSH on an internal database cluster five minutes later, it programmatically cross-references these logs, synthesizes the lateral movement timeline, and instantaneously elevates the event into a critical, high-fidelity security incident.

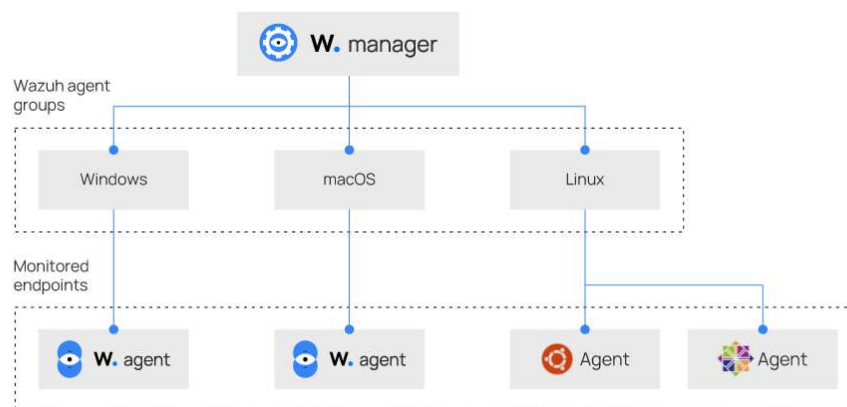
2.5.2 The Operational Bottleneck: Alert Fatigue

While SIEM systems are highly effective at consolidating enterprise telemetry, their reliance on expansive signature rulesets introduces a critical operational vulnerability known as **Alert Fatigue**. Security Operations Centers (SOCs) are routinely inundated with thousands of security notifications per day. A significant portion of these alerts consists of low-severity indicators, benign network anomalies, or false positives—such as a legitimate administrative script triggering an alert for suspicious privilege execution.

When human security analysts are subjected to a continuous, unmanaged stream of high-volume alerts, cognitive saturation occurs. The operational impact of alert fatigue manifests in two primary risks:

- **The "Noise" Saturation Effect:** Critical, high-fidelity threat indicators representing active Advanced Persistent Threat (APT) intrusions or zero-day lateral movements become buried in the massive volume of benign log data.
- **Delayed Response Windows:** Analysts become desensitized to recurring warnings, leading to missed notifications or extended Mean Time to Acknowledge (MTTA) and Mean Time to Remediate (MTTR) metrics, which grants malicious actors extended dwell time inside the corporate domain. Resolving this critical bottleneck requires integrating upstream automation and orchestration pipelines (such as SOAR architectures) to programmatically filter the noise and isolate high-fidelity incidents.

2.5.3 Wazuh Architecture: The Manager-Agent Model



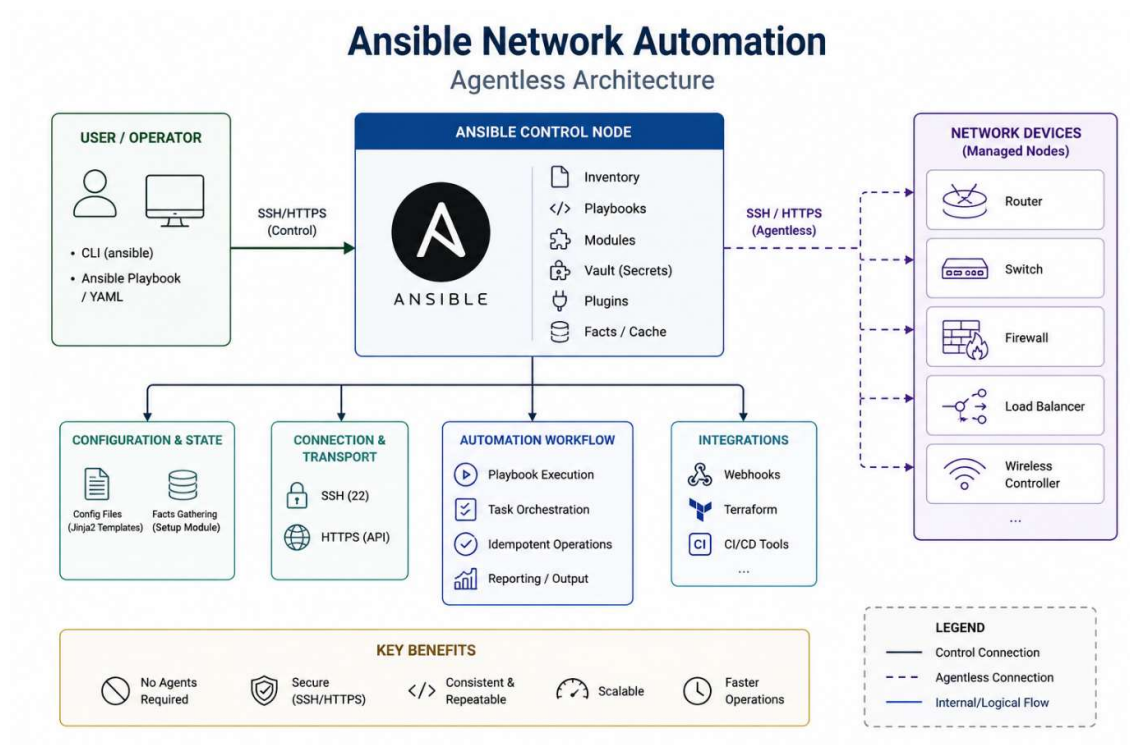
In modern enterprise security deployments and open-source virtualized architectures, **Wazuh** functions as an open-source XDR/SIEM platform integrating Host Intrusion Detection (HIDS), Security Monitoring, Vulnerability Detection, and File Integrity Monitoring (FIM) capabilities. Rather than deploying a full-scale standalone EDR suite, Wazuh evolves traditional SIEM models by utilizing a decentralized, highly scalable **Manager-Agent architecture** to achieve granular host-level and network-level visibility across the enterprise fabric.

- **The Wazuh Agent:** This is a lightweight, multi-platform system daemon deployed directly onto monitored infrastructure endpoints (such as Windows workstations, Linux virtual servers, or corporate domain controllers). The agent operates in user space with minimal infrastructure overhead, executing continuous host-level telemetry tracking. Its internal tasks include active system log monitoring, **File Integrity Monitoring (FIM)** to detect unauthorized modifications to critical system files or configurations, rootkit detection scans, and local software inventory gathering for security analysis.
- **Network Demarcation and Port Allocation:** The communication interface between the distributed agents and the centralized manager is strictly governed by distinct network port assignments to preserve control-plane integrity and data-plane isolation:
 - **Port 1515 (TCP):** Dedicated exclusively to the initial **Agent Enrollment and Registration** phase. This is a one-time transaction where a new endpoint requests entry into the Wazuh domain, validates its credentials, and securely receives its unique cryptographic pre-shared key.
 - **Port 1514 (TCP):** Dedicated to ongoing, continuous **Agent Communication**. All real-time telemetry, log streams, FIM alerts, and system state data are transmitted over this channel. Cryptographic protection and mutual validation are handled natively by Wazuh's custom communication protocol and authentication keys.
 - **Port 1514 (UDP or TCP - Syslog Collector):** Utilized by the manager to ingest standard raw Syslog telemetry directly from network hardware assets that do not support agent installations, such as perimeter pfSense firewalls and core enterprise switches. The underlying transport layer protocol (UDP or TCP) is dynamic and depends entirely on the manager's explicit configuration settings.
- **The Wazuh Manager:** This acts as the centralized control plane and analytical hub of the security architecture. The manager is responsible for parsing, decoding, and analyzing the incoming telemetry streams sent by distributed agents or syslog sources. It passes the data through its centralized **Analysis Engine**, which evaluates the incoming decoded events against an extensive matrix of pre-configured detection rules. These rules are enriched with explicit MITRE ATT&CK technique mappings to contextually classify threat behaviors. If an event triggers a rule match above a defined severity threshold, or matches identified software vulnerabilities during the

Vulnerability Detection cycle (where host software versions are cross-referenced with continuous CVE databases), the manager generates a structured security alert payload.

- **Storage and Visualization: Wazuh Indexer and Dashboard:** Once the analysis engine generates an alert, the structured JSON payload is transmitted downstream to the storage and presentation layers. Modern deployments utilize the **Wazuh Indexer**—a highly optimized, distributed, full-text search and analytics engine derived from open-source OpenSearch technology. The indexer organizes the raw alert vectors into chronological, time-series indices. This foundational data layer is then visually rendered by the **Wazuh Dashboard**, providing security operations teams with real-time graphical interfaces, compliance auditing views, and actionable threat forensic mining capabilities.

2.6 Network Automation — Ansible



Traditional enterprise network administration has historically relied on manual, hop-by-hop configurations executed via command-line interfaces (CLI) or graphical web views. While

functional for legacy scale, this human-driven methodology introduces significant operational risks, including configuration drift, extended provisioning windows, and catastrophic human-induced outages. Network automation mitigates these vulnerabilities by replacing manual configuration with programmatic, repeatable, and scalable software-driven orchestration mechanisms.

2.6.1 Infrastructure as Code (IaC) Core Concepts

At the architectural center of modern network automation is the paradigm of **Infrastructure as Code (IaC)**. Originally pioneered in cloud computing environments, IaC treats hardware and software infrastructure configurations with the identical software development lifecycle engineering methodologies applied to application source code.

In an enterprise network domain, IaC enforces several foundational principles:

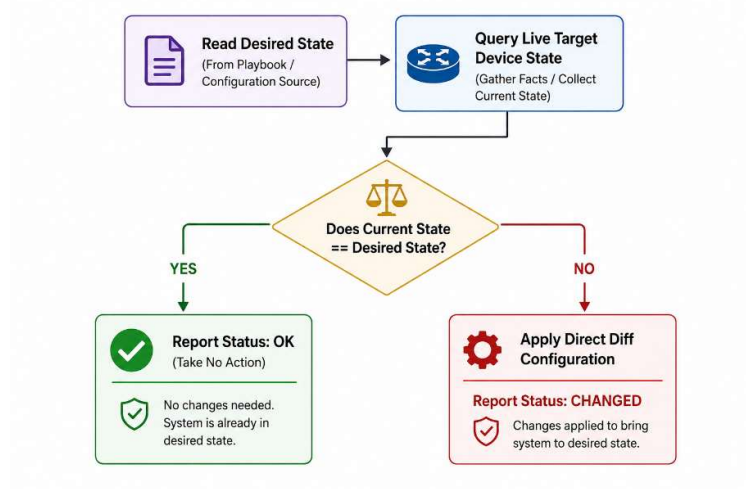
- **Declarative vs. Imperative Paradigms:** Instead of writing imperative execution steps (e.g., explicitly instructing a router to execute a sequence of CLI commands), administrators define the infrastructure's **desired state** in static, human-readable structured data files (typically formatted in YAML or JSON). The automation engine evaluates this file and programmatically derives the exact configuration steps required to transition the network node to that target state.
- **Single Source of Truth:** Network designs, security policies, access-control lists, and VLAN interfaces are abstracted into a centralized repository. Version control systems (such as Git) host these files, allowing changes to be peer-reviewed, audited, and rolled back systematically.
- **Elimination of Configuration Drift:** Continuous application of IaC playbooks ensures that out-of-band, manual modifications made directly to localized hardware interfaces are programmatically detected and automatically overwritten to maintain alignment with the master repository blueprint.

2.6.2 The Core Operational Principle: Idempotency

The definitive characteristic that distinguishes advanced automation frameworks from basic scripting utilities (such as Python Netmiko or Bash loops) is the mathematical concept of **Idempotency**. An automation task or module is classified as idempotent if the outcome of

executing it multiple times consecutively yields the identical systemic state as executing it exactly once, without introducing unintended side effects or redundant operations.

When an idempotent framework like Ansible executes a configuration change across enterprise target nodes, it executes a rigorous, multi-step state verification workflow:



Traditional procedural scripts lack this capability; executing a primitive script to append a specific rule to a firewall interface multiple times will result in duplicate lines or configuration errors. In sharp contrast, an idempotent module first inspects the target device's running configuration. If the defined state already exists, the engine logs an ok status and bypasses execution. If a discrepancy is identified, it executes only the explicit delta configuration required to bring the device into compliance, logging a changed status. This guarantees absolute predictability and safety during large-scale production network deployments.

2.6.3 Ansible Architecture and Component Engineering

Ansible is widely adopted as an industry-standard configuration management platform due to its inherently scalable, **agentless architecture**. Unlike traditional server management systems that require a persistent software daemon or agent to be actively running on the target machine, Ansible executes commands completely externally. It pushes configuration payloads down to target nodes utilizing standard, native communication protocols—predominantly **SSH** (Secure Shell) for standard Unix/Linux endpoints and CLI-based enterprise routing hardware, and **NETCONF** (Network Configuration Protocol) or REST

APIs for modern multi-vendor programmable platforms. This makes it uniquely qualified for network devices whose locked operating systems prohibit the installation of third-party monitoring daemons.

The framework's execution model is organized into three core interdependent structural components:

The Ansible Inventory

The inventory file (typically constructed in INI or YAML syntax) acts as the comprehensive registry of the automation infrastructure. It maps the hostnames, fully qualified domain names (FQDNs), or IP addresses of every manageable network asset to specific logical groups (e.g., segregating infrastructure components into `core_switches`, `perimeter_firewalls`, or `edge_routers`). Furthermore, the inventory handles variables across distinct hierarchical scopes, allowing engineers to bind specific management parameters—such as explicit IP addresses, unique BGP Autonomous System Numbers (ASNs), or custom interface configurations—to individual nodes or global groups.

Ansible Playbooks

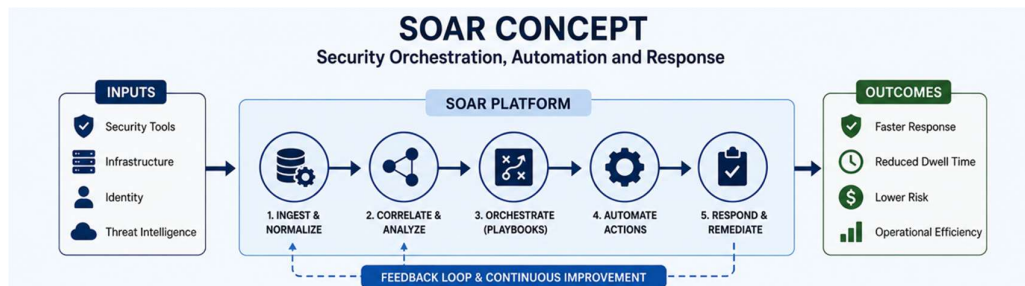
Playbooks serve as the absolute configuration blueprints within the IaC pipeline. Written in strict YAML notation, a playbook is composed of one or more "plays," which map specific target inventory host groups to a logical sequence of execution blocks called **Tasks**. Each individual task invokes a discrete, specialized **Ansible Module** (e.g., utilizing `cisco.ios.ios_l3_interfaces` for Cisco gear or `fortinet.fortios.fortios_configuration` for security gateways). Modules abstract the complex vendor-specific syntactic commands behind standardized variables, executing the underlying code programmatically on the execution controller and rendering the device configuration dynamically.

Ansible Roles

As enterprise automation architectures scale out across complex multi-site topologies, maintaining flat playbook files introduces severe modularity bottlenecks. **Ansible Roles** enforce an organized, standardized file directory abstraction layer that enables the modular segregation of configuration logic into distinct, reusable operational blocks. A structural role separates variables (`vars/`), static parameters (`defaults/`), target templates (`templates/`), execution

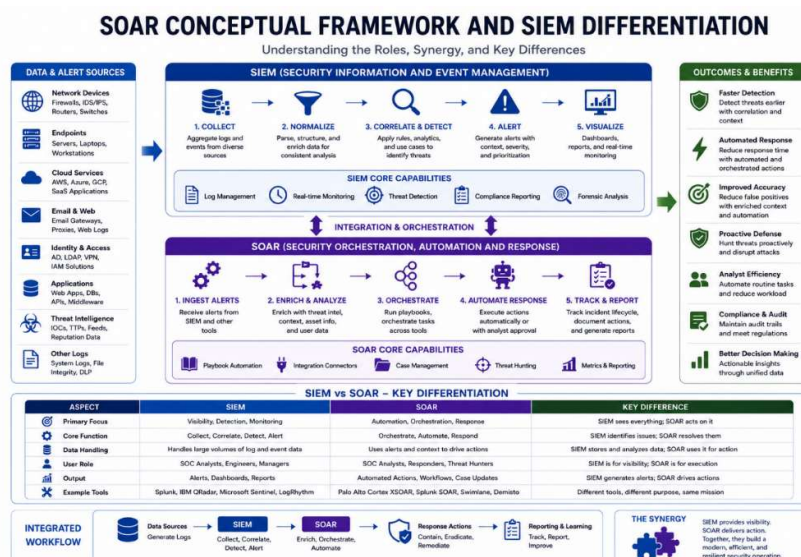
tasks (tasks/), and state-driven triggers (handlers/) into distinct isolated subdirectories. This decoupling enables an organization to build generic, reusable network service frameworks (such as an OSPF provisioning role or a baseline hardening blueprint) that can be invoked dynamically across different devices and scopes, enforcing structural maintainability across the global enterprise fabric.

2.7 Security Orchestration, Automation & Response (SOAR)



While traditional Security Operations Centers (SOCs) rely heavily on SIEM platforms to aggregate logs and detect sophisticated threat vectors, detection represents only the initial phase of cybersecurity defense. Once a high-fidelity alert is generated, security analysts must execute critical containment, verification, and remediation procedures. Executing these workflows manually across disparate multi-vendor security consoles drastically increases the Mean Time to Remediate (MTTR). Security Orchestration, Automation, and Response (SOAR) frameworks address this operational bottleneck by programmatically integrating security architectures into automated, event-driven incident response pipelines.

2.7.1 SOAR Conceptual Framework and SIEM Differentiation



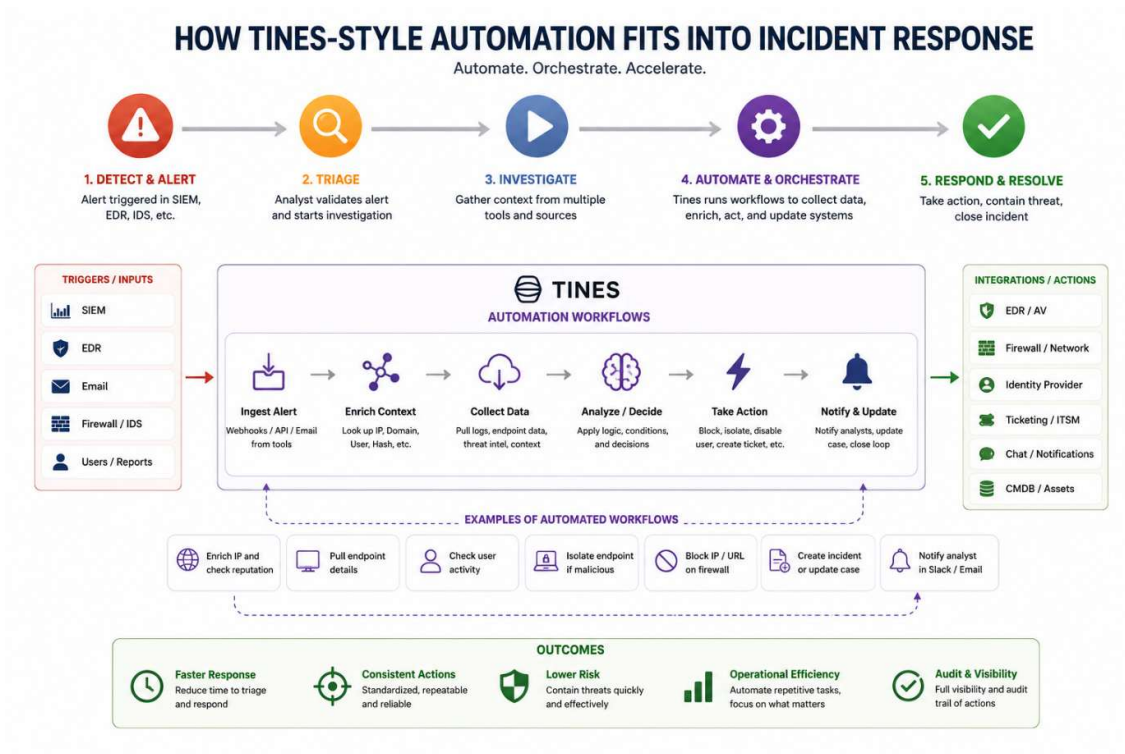
To build a highly resilient, autonomous security architecture, network engineers must clearly differentiate between the structural functions of SIEM and SOAR technologies. While they operate synergistically within a unified defensive ecosystem, their primary operational objectives are fundamentally distinct:

- SIEM (The Analytical Core - "The Eyes and Brain"):** The primary mission of a SIEM system is comprehensive visibility, threat detection, and historical log retention. It ingests heterogeneous telemetry streams, normalizes the data, evaluates it against defined correlation matrices, and flags anomalies or known threat behaviors. A SIEM, however, is inherently passive; it excels at identifying that an enterprise asset is undergoing a brute-force or lateral movement attack, but it does not natively interface with downstream network controls to halt the intrusion.
- SOAR (The Operational Engine - "The Hands and Muscle"):** SOAR extends the security pipeline by introducing programmatic actionability. It ingests security alerts directly from the upstream SIEM (or external threat intelligence feeds) and orchestrates automated workflows across the organization's entire technological fabric via Application Programming Interfaces (APIs). SOAR architectures leverage digitized **Playbooks**—formalized, step-by-step conditional logical workflows—to systematically execute automated remediation steps without requiring constant human intervention.

Table 2.1: Structural Comparison Between SIEM and SOAR Frameworks

Evaluation Metric	Security Information & Event Management (SIEM)	Security Orchestration, Automation & Response (SOAR)
Primary Objective	Data aggregation, parsing, parsing enrichment, log correlation, and threat visibility.	Workflow orchestration, incident triage mitigation, and automated system remediation.
Core Input Layer	Raw, unstructured machine logs, system metrics, and transport syslog streams.	High-fidelity security alerts, parsed JSON metadata payloads, and webhook triggers.
Operational State	Analytical and observation-driven (Passive warning generation).	Executional and operational-driven (Active enforcement execution).
Typical Outputs	Unified dashboards, forensic timeline graphs, and security event alerts.	Dynamic API blocks, firewall rule alterations, endpoint isolations, and ticket routing.

2.7.2 Low-Code/No-Code Workflow Automation — Tines



In contemporary agile enterprise infrastructures and software-defined security architectures, traditional heavy-footprint enterprise SOAR utilities are increasingly supplemented or

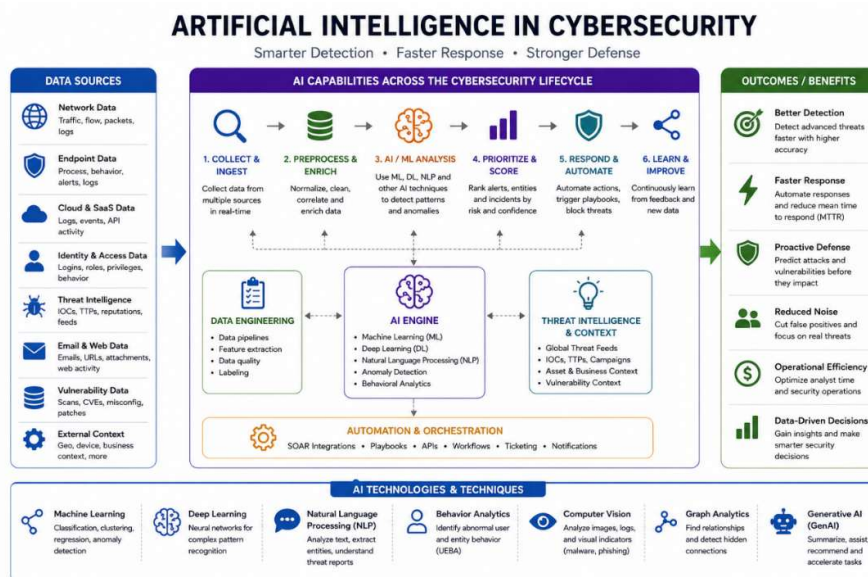
replaced by modern, lightweight **Low-Code/No-Code Workflow Automation Platforms** such as **Tines**. These platforms function as highly adaptable, event-driven integration layers, abstracting complex backend script mechanics into scalable, node-based logical pipelines that are optimized for rapid deployment within an incident response framework.

The integration of a Tines orchestration engine into a modern SOC architecture introduces a standardized, programmatic lifecycle for automated incident mitigation:

1. **Event Ingestion via Dynamic Webhooks:** When a centralized analysis engine (such as a Wazuh Manager) triggers an alert that crosses a predefined critical risk threshold, it does not merely log the event. The manager executes an upstream HTTP POST request, streaming a structured JSON data payload directly to a secure, designated **Webhook Listener Node** hosted on the automation platform.
2. **Programmatic Telemetry Enrichment:** The automation engine receives the raw alert data (containing indicators of compromise like a malicious external IP address or an anomalous internal user account) and programmatically queries auxiliary intelligence services via REST APIs. The engine can query reputation databases (such as VirusTotal or AbuseIPDB) to ascertain the threat score of the external vector, consolidating all contextual metadata within milliseconds.
3. **Conditional Logic Evaluation:** Utilizing clean graphical decision nodes, the workflow evaluates the consolidated data against rigorous engineering criteria. For example, the pipeline determines: *Is the external threat score greater than a critical threshold, and does the target asset host critical database systems?*
4. **Multi-Vendor Orchestrated Remediation:** If the conditions match a validated intrusion signature, the automation engine parallelizes native API-driven actions across multiple infrastructure layers simultaneously:
 - **Network Security Perimeter:** It sends an authenticated API call to the perimeter firewall (such as Netgate pfSense), appending the malicious source IP address to a dynamic alias blocklist to instantaneously sever WAN-to-LAN transit.
 - **Configuration Enforcement:** It can call an automation controller (such as Ansible) to execute an emergency playbook, dynamically modifying VLAN assignments or isolating a compromised host machine from the internal corporate core.

- **Incident Management & Collaboration:** Simultaneously, the platform generates a structured tracking ticket within an IT service management tool (e.g., Jira or ServiceNow) and broadcasts a formatted cryptographic event synopsis to the security team's communication channels (e.g., Slack or Microsoft Teams) detailing the precise defensive actions taken programmatically.

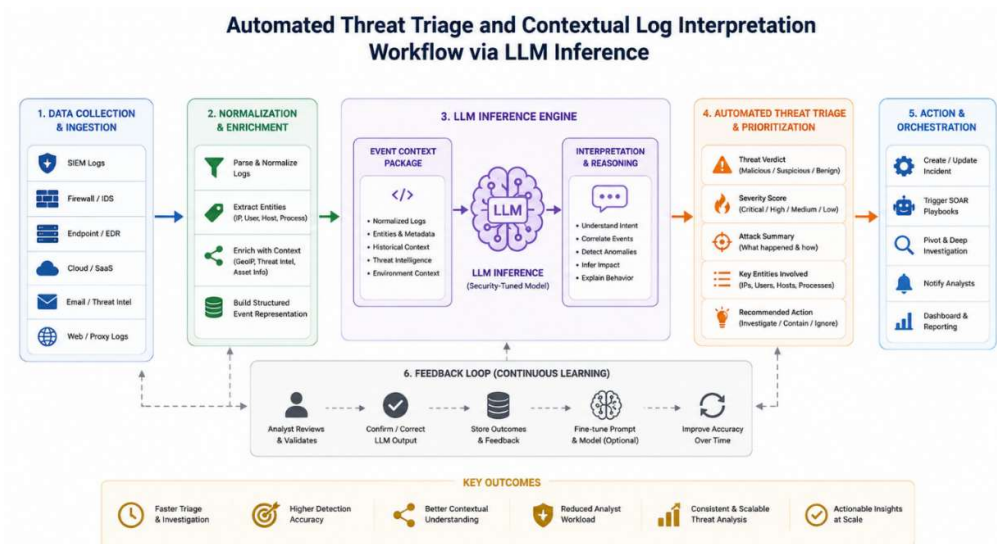
2.8 Artificial Intelligence in Cybersecurity



The exponential increase in the volume, velocity, and sophistication of contemporary cyber threats has exposed the inherent limitations of traditional, purely signature-based defense paradigms. Advanced Persistent Threats (APTs), polymorphic malware, and automated multi-stage external probes bypass traditional security rules by altering their structural attributes while maintaining malicious intent. To address this operational gap, modern defensive network architectures increasingly incorporate Artificial Intelligence (AI) and Large Language Models (LLMs) to achieve autonomous anomaly detection, automated

contextual log analysis, and intelligent predictive processing.

2.8.1 LLMs for Log Analysis and Automated Threat Triage



Within modern Security Operations Centers (SecOps), the execution of threat triage—the systematic process of ingesting, screening, and prioritizing incoming alerts—is severely hindered by the sheer volume of unstructured machine telemetry. Large Language Models (LLMs) transform this workflow by introducing a cognitive abstraction layer capable of parsing complex, vendor-specific event payloads (such as cryptic Linux kernel logs, database queries, or pfSense firewall states) into deterministic intelligence.

The integration of LLMs inside an automated defensive pipeline establishes two main operational advantages:

- **Contextual Alert Enrichment:** By forwarding a parsed JSON alert blob from a SIEM platform directly to an LLM via API, the model contextually cross-references the technical indicators against global threat taxonomies. It infers the attacker’s underlying operational objective, maps the behavior to known methodologies (such as the MITRE ATT&CK matrix), and evaluates whether the observed sequence constitutes an isolated anomaly or an active, multi-stage network intrusion.

- Intelligent Triage Verification:** Rather than relying exclusively on fixed severity levels, the LLM dynamically conducts threat triage by scoring the alert based on real-time operational context. The model generates semantic summaries, assigns a dynamic fidelity level to the threat, and isolates high-priority alerts from benign baseline noise. Furthermore, it accelerates remediation timelines by instantly drafting specific, tailor-made configuration tasks—such as specific command lines or Ansible orchestration structures—to neutralize the verified compromise.

2.8.2 Architectural Trade-Offs: Local vs. Cloud-Based AI

When architecting an AI-driven security ecosystem, network security engineers must navigate a critical design dichotomy: deploying a localized on-premise model configuration or integrating a high-performance cloud-based inference ecosystem. Both deployment models introduce distinct operational and structural trade-offs:

Table 2.2: Architectural Matrix of Local vs. Cloud-Based AI Deployment

Engineering Metric	Local LLM Deployment (e.g., Ollama / On-Premise)	Cloud-Based AI Inference (e.g., Groq, OpenAI, Gemini)
Data Privacy & Control	Complete data isolation; sensitive metadata remains inside the local secure boundary.	Inherent exposure risk; sensitive data leaves the perimeter via external web APIs.
Computational Speed	Highly constrained by local hardware resource availability (GPU/VRAM limitations).	Ultra-low latency and deterministic response times powered by dedicated hardware.
Operational Overhead	High; requires ongoing localized hardware maintenance and complex engineering optimization.	Minimal; abstract infrastructure management to a scalable cloud service model.

The Shared Privacy Vulnerability of Cloud-Based AI Ecosystems

From a strict data sovereignty and security compliance perspective, all cloud-based AI platforms—including OpenAI (ChatGPT), Google (Gemini), Anthropic (Claude), and Groq—share an identical architectural vulnerability: they require security telemetry to leave the local network perimeter. Security logs are highly sensitive, containing internal IP routing

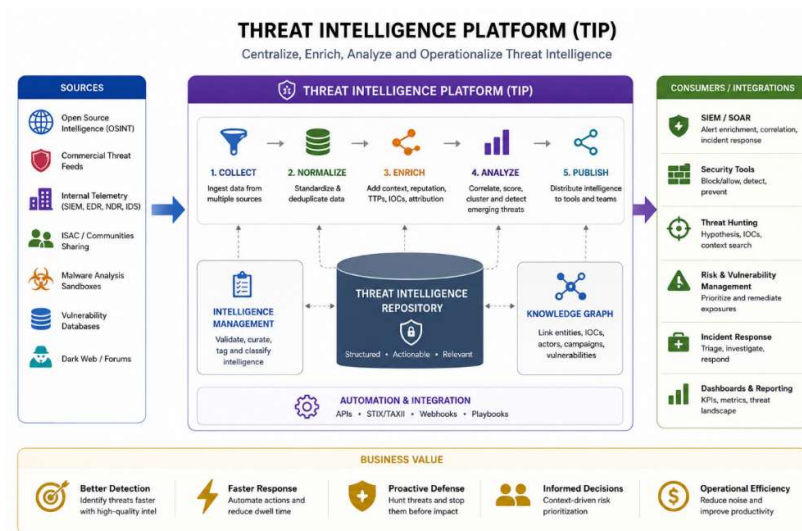
configurations, hostnames, administrator usernames, and software versioning strings. Processing these unstructured payloads over external cloud connections inherently introduces risks of data exposure, external archiving, and third-party dependency.

2.8.3 Academic and Industry Implementations

The deployment of Large Language Models for automated security orchestration is heavily validated by prominent academic frameworks and pioneering enterprise industry implementations:

- **Industry Paradigms (Google Sec-PaLM & Microsoft Security Copilot):** In commercial security domains, hyper-scale providers have demonstrated the viability of specialized cyber-centric LLMs. Google's **Sec-PaLM** (a model fine-tuned explicitly on security telemetry and global threat research) and Microsoft's **Security Copilot** actively demonstrate how LLMs can ingest massive streams of enterprise logs to automatically synthesize incident timelines and reverse-engineer malicious scripts in real time.
- **Academic Architectures (LLM-Based Autonomous Agents):** Recent academic research focus has shifted toward building "Autonomous SOC Agents." These peer-reviewed frameworks demonstrate how an LLM, when given access to isolated external tools via secure webhooks, can operate as a closed-loop incident responder. Academic models prove that combining an analytical engine (SIEM) with a workflow controller (SOAR) and a cognitive reasoning unit (LLM) allows networks to autonomously identify, analyze, and apply direct firewall rules to block ongoing external attacks without human latency.

2.9 Threat Intelligence Platforms (TIP)



Threat detection and incident response cannot operate effectively in isolation. Relying solely on internal network telemetry and locally generated security logs leaves organizations vulnerable to zero-day exploits, coordinated cyber campaigns, and Advanced Persistent Threat (APT) groups. To transition from a reactive security posture to proactive cyber defense, modern Security Operations Centers (SOCs) implement Cyber Threat Intelligence (CTI).

A Threat Intelligence Platform (TIP) serves as the centralized intelligence layer that aggregates, normalizes, enriches, correlates, and distributes threat intelligence collected from multiple external and internal sources. By integrating continuously updated threat data into existing security controls, a TIP enhances detection accuracy, accelerates incident response, and enables automated security operations across heterogeneous enterprise environments.

2.9.1 Taxonomy of Cyber Threat Intelligence and Indicators of Compromise (IoCs)

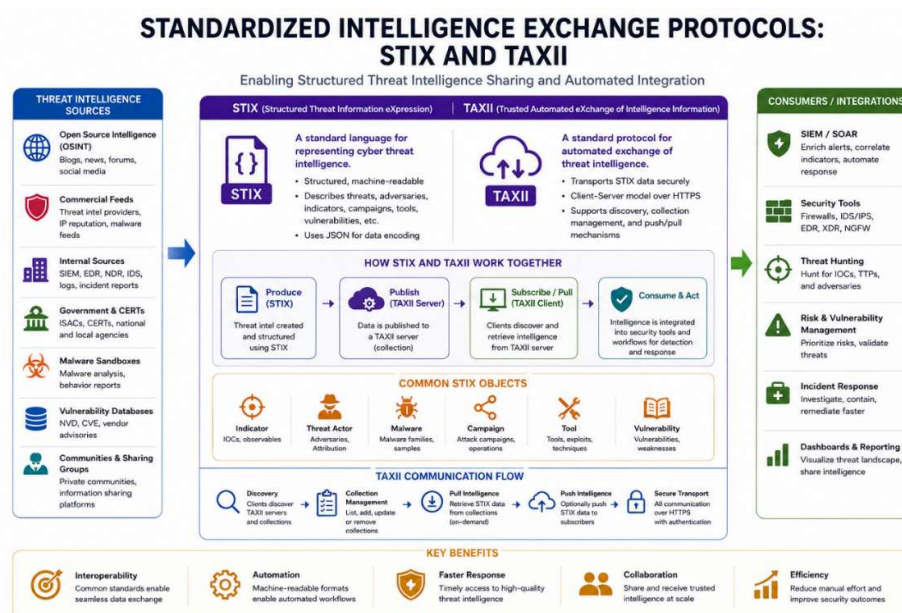
Cyber Threat Intelligence (CTI) is commonly classified into three complementary intelligence tiers, each supporting a different level of organizational decision-making and security operations:

- **Strategic Intelligence:** Provides high-level, non-technical analysis concerning adversary motivations, geopolitical developments, emerging cyber risks, and industry-

specific threat trends. This intelligence supports executive decision-making, long-term cybersecurity strategy, enterprise risk management, and security investment planning.

- **Operational Intelligence:** Focuses on active cyber campaigns and adversary behavior. It provides insight into the Tactics, Techniques, and Procedures (TTPs) employed by threat actors, which are commonly mapped using the MITRE ATT&CK framework. This intelligence enables security teams to understand attacker methodologies and improve detection engineering.
- **Tactical Intelligence:** Consists of highly technical Indicators of Compromise (IoCs) that can be directly consumed by security technologies. Typical IoCs include malicious IP addresses (IPv4 and IPv6), malicious domain names (FQDNs), suspicious URLs, email addresses, and cryptographic file hashes such as SHA-256 (and legacy MD5 values). Tactical intelligence provides machine-readable data that enables firewalls, intrusion detection systems, SIEM platforms, and endpoint security solutions to automatically identify and block known malicious activity.

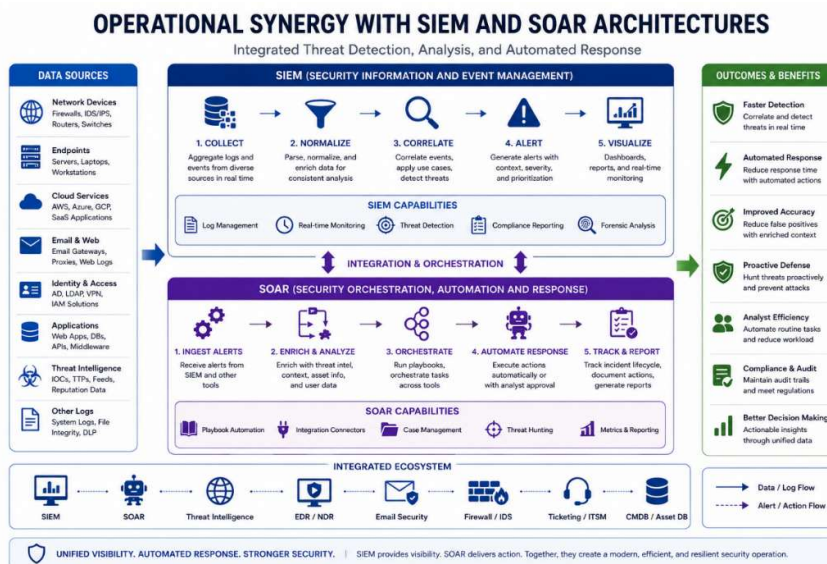
2.9.2 Standardized Intelligence Exchange Protocols: STIX and TAXII



To facilitate interoperability among heterogeneous cybersecurity products, the industry has adopted standardized intelligence-sharing protocols developed by the Organization for the Advancement of Structured Information Standards (OASIS). These standards enable automated, vendor-independent exchange of cyber threat intelligence.

- **STIX (Structured Threat Information eXpression):** A structured, JSON-based language for representing cyber threat intelligence. It models relationships among cyber threat entities—including threat actors, malware families, campaigns, vulnerabilities, indicators, infrastructure, and recommended Courses of Action (CoAs)—allowing security platforms to exchange intelligence using a consistent and machine-readable format.
- **TAXII (Trusted Automated eXchange of Intelligence Information):** A RESTful application-layer protocol operating over HTTPS that defines how STIX intelligence is securely exchanged between systems. TAXII enables Threat Intelligence Platforms to automatically retrieve threat feeds from external intelligence providers and distribute normalized intelligence to downstream security technologies such as SIEM, SOAR, firewalls, and Endpoint Detection and Response (EDR) platforms.

2.9.3 Operational Synergy with SIEM and SOAR Architectures

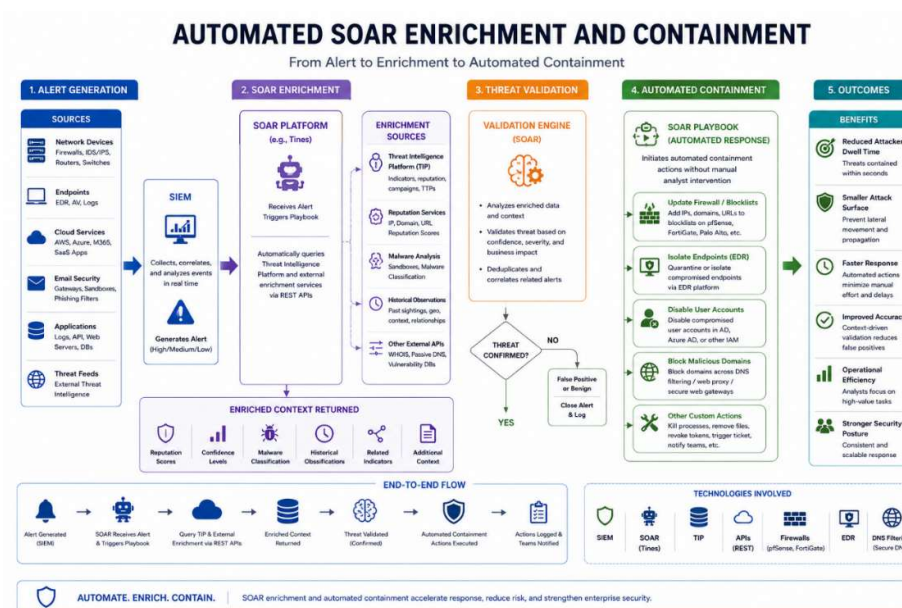


A centralized Threat Intelligence Platform—such as MISP (Malware Information Sharing Platform & Threat Sharing)—acts as the organization's primary repository for cyber threat intelligence. By integrating external reputation and enrichment services, including VirusTotal and AbuseIPDB, the platform continuously enriches collected intelligence and distributes validated Indicators of Compromise throughout the Security Operations Center (SOC).

Proactive SIEM Correlation

The TIP continuously synchronizes updated IoC datasets with the Security Information and Event Management (SIEM) platform (e.g., Wazuh). As endpoints, network devices, firewalls, and servers forward security logs to the SIEM, its correlation engine compares incoming events against the latest threat intelligence repository. When an event matches a known malicious IoC, the SIEM automatically elevates the event into a high-confidence security alert, significantly reducing false positives while improving detection accuracy.

Automated SOAR Enrichment and Containment



Following alert generation, the Security Orchestration, Automation, and Response (SOAR) platform (e.g., Tines) automatically queries the Threat Intelligence Platform and external enrichment services through REST APIs. Additional contextual information—including reputation scores, confidence levels, malware classifications, and historical observations—is retrieved to validate the detected threat.

If the threat is confirmed, the SOAR playbook initiates automated containment procedures without requiring manual analyst intervention. These actions may include dynamically updating firewall address groups or blocklists on platforms such as pfSense or FortiGate, isolating compromised endpoints through Endpoint Detection and Response (EDR) systems, disabling compromised user accounts, or blocking malicious domains across DNS filtering services. This automated response minimizes attacker dwell time, reduces the attack surface, and prevents lateral movement before the threat can propagate throughout the enterprise network.

Chapter 3 — System Design

3.1 Design Overview

To architect an enterprise-grade autonomous security framework, the system is engineered around a comprehensive **3-Layer Architecture Model**: the **Infrastructure Layer**, the **Automation Layer**, and the **Intelligence Layer**.

Rather than treating routing, orchestration, and threat analysis as isolated operational silos, this design fuses them into a continuous, closed-loop pipeline. If a threat penetrates the perimeter, it is immediately detected, analyzed, contextually verified, and programmatically mitigated without human latency.

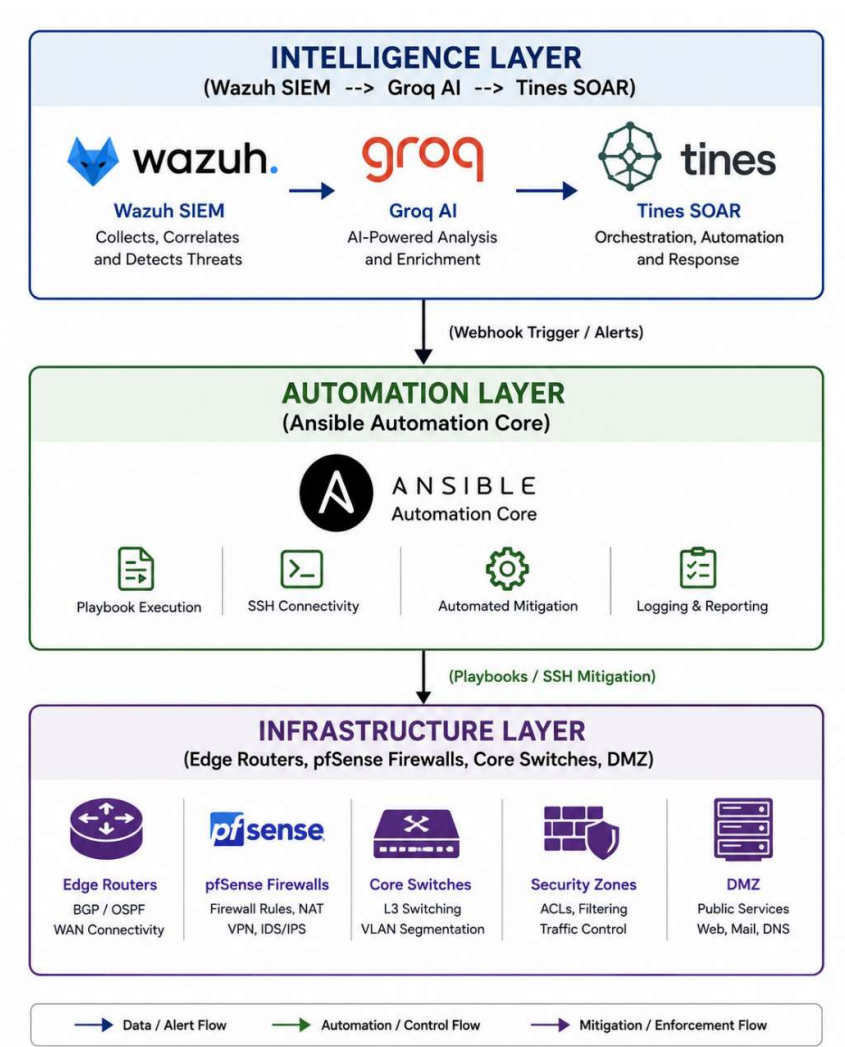


Figure 3.1: High-Level Conceptual 3-Layer Architecture Matrix.

3.1.1 The Infrastructure Layer (The Foundation)

The Infrastructure Layer serves as the physical and logical foundation of the enterprise network, managing raw data transit, stateful perimeter security, and workload isolation. As visually represented in the upper sections of the system topology (the green and purple zones in image_7d343d.jpg), this layer is responsible for baseline connectivity:

- **Perimeter and Routing Control:** Governed by the Edge Router (R-isp) which manages multi-homed WAN connections and handles raw traffic ingress from the Internet cloud.
- **Core Security and Segmentation:** Internal core networks are segmented via Layer-3 routing domains (R1, R2, R3) and protected by stateful Next-Generation Firewalls (A-pfSense, fw-HQ, fw-b). These firewalls isolate internal computing zones, host specialized application containers (Docker), and maintain persistent, encrypted Site-to-Site IPsec VPN tunnels to bind corporate nodes together.
- **Local Ingress/Egress Switching:** Managed through distributed switching matrices (sw1, sw2, sw3) that directly terminate endpoint devices, broadcast domains, and localized enterprise VLANs.

3.1.2 The Automation Layer (The Enforcer)

The Automation Layer represents the execution branch of the architecture, responsible for programmatic network provisioning, policy enforcement, and real-time threat containment. Located within the dedicated management sector of the network topology (the yellow-green container in image_7d343d.jpg), this layer introduces agentless orchestration:

- **Ansible Control Core:** Centered around a dedicated **Ansible Control Node** that communicates out-of-band with network infrastructure components over secured management planes.
- **Dynamic Playbook Execution:** Rather than requiring manual administrative CLI intervention during a network breach, the Automation Layer receives tactical instructions (TO TAKE ACTION) from the upstream intelligence engine. It instantly compiles and deploys targeted YAML playbooks to dynamically alter firewall state

tables, null-route malicious assets on edge switches, or isolate compromised containers.

3.1.3 The Intelligence Layer (The Cognitive Brain)

The Intelligence Layer constitutes the cognitive reasoning engine of the architecture. It is responsible for continuous log telemetry aggregation, behavioral threat detection, high-speed cognitive triage, and orchestration workflow control. This layer is consolidated within an isolated security operations enclave (the dark green container in image_7d343d.jpg):

- **Telemetry Aggregation (Wazuh SIEM):** The **Wazuh Manager** continuously ingests security logs, system events, and integrity metrics from endpoints and firewalls across the entire infrastructure.
- **Cognitive Log Triage (Groq AI Engine):** High-priority unstructured security alerts are immediately pushed to the **Groq AI Inference Pipeline**. Leveraging Language Processing Unit (LPU) specialized hardware speed, it decodes and triages complex log sequences within milliseconds, filtering out noise and confirming valid compromises.
- **Orchestration and Alerting (Tines SOAR):** Upon receiving validated threat intelligence, **Tines SOAR** triggers automated playbooks. It programmatically dispatches structural mitigation commands directly to the Ansible Automation Core, while simultaneously broadcasting instant rich-text security notifications and compliance logs to enterprise channels like **Telegram** and **Gmail**.

3.2 Network Topology

3.2 Network Topology

The network infrastructure is modeled as a multi-site enterprise architecture emulated within PNetLab. It is logically and physically segmented into four functional zones distinguished by color-coded containers in **Figure 3.2**.

(Insert the full topology image here)

Figure 3.2: Multi-Site Enterprise Infrastructure and Autonomous SOC Topology.

- **Perimeter & WAN Zone (Green Container):**
 - Governed by the core ISP router (R-isp) simulating public internet routing.
 - Distributes point-to-point /30 public subnets to all edge nodes (10.10.10.0/30, 10.10.10.4/30, and 10.10.10.8/30).
 - Includes a dedicated boundary asset (dos attack) to simulate live external threat vectors.
- **Security & Routing Core (Purple Container):**
 - **Left Wing:** Anchored by R1 and A-pfSense firewall to protect local Docker application workloads and switching fabrics (sw1).
 - **Center Headquarters (HQ):** Driven by R2 and the primary enterprise firewall (fw-HQ) controlling the central switching matrix (sw2).
 - **Right Wing:** Represents a remote branch secured by R3 and fw-b firewall managing branch access switches (sw3).
 - **Overlay Layer:** All perimeters are interconnected via secure, stateful Site-to-Site cryptographic tunnels to ensure encrypted corporate transit.
- **Intelligence Enclave / SOC (Dark Green Container):**
 - Houses **Wazuh SIEM Manager** for central log aggregation and correlation from all enterprise assets.
 - Integrates with **Groq AI Engine** for sub-second cognitive log triage and threat analysis.
 - Utilizes **Tines SOAR** to orchestrate response playbooks and dispatch external alerts via **Telegram** and **Gmail**.
- **Automation Enclave (Yellow Container):**
 - Dedicated to Out-of-Band (OOB) infrastructure configuration management.
 - Centered around an agentless **Ansible Control Node** connected directly into the Headquarters switching fabric (sw2) to execute programmatic, zero-latency threat isolation and containment.

3.3 IP Addressing Scheme

To guarantee deterministic routing paths, effective micro-segmentation, and conflict-free traffic flows across the multi-site infrastructure, a structured IPv4 addressing matrix has been deployed.

Table 3.1 details the comprehensive IP allocation scheme for all network interfaces, core transit links, internal local area networks (LANs), and security management enclaves.

Table 3.1: Comprehensive Enterprise IP Addressing and Interface Matrix

Device Name	Interface	Connected To / Segment	IP Address / Subnet	Default Gateway
R-isp	e0/0	Left WAN Transit Link	10.10.10.1/30	N/A
	e0/1	HQ WAN Transit Link	10.10.10.5/30	N/A
	e0/2	Branch WAN Transit Link	10.10.10.9/30	N/A
R1	e0/0	R-isp (e0/0)	10.10.10.2/30	10.10.10.1
	e0/2	A-pfSense (e0)	172.16.1.1/29	N/A
A-pfSense	e0	R1 (e0/2)	172.16.1.2/29	172.16.1.1
	e1	HQ LAN Switch (sw1)	10.10.10.1/16	N/A
R2	e0/0	R-isp (e0/1)	10.10.10.6/30	10.10.10.5
	e0/1	fw-HQ (e0)	172.16.2.1/29	N/A
fw-HQ	e0	R2 (e0/1)	172.16.2.2/29	172.16.2.1
	e1	Core HQ Switch (sw2)	10.10.20.1/16	N/A
R3	e0/0	R-isp (e0/2)	10.10.10.10/30	10.10.10.9
	e0/1	fw-b (e0)	172.16.3.1/29	N/A
fw-b	e0	R3 (e0/1)	172.16.3.2/29	172.16.3.1
	e1	Branch Switch (sw3)	10.10.30.1/16	N/A
Wazuh SIEM	eth0	Services DMZ Layer (VLAN 50)	172.16.50.10/24	172.16.50.1
Docker Host	eth0	Containerized App Zone (VLAN 50)	172.16.50.20/24	172.16.50.1
Ansible Node	eth0	Out-of-Band Mgmt (VLAN 99)	172.16.99.10/24	172.16.99.1
HQ Users (VPCs)	eth0	Internal Corporate Users (VLAN 10)	DHCP Range	10.10.20.1

3.4 Tools & Technologies Used

The selection of each technology within this project was driven by a combination of technical suitability, cost accessibility, interoperability, and alignment with real-world enterprise deployment patterns. The following subsections justify each architectural choice.

3.4.1 PNetLab — Simulation Environment

PNetLab was selected as the primary network simulation platform over alternatives such as GNS3 or Cisco Packet Tracer for three specific reasons. First, PNetLab natively supports the execution of Cisco IOL (IOS on Linux) images, enabling authentic Cisco IOS behavior without requiring physical hardware. Second, its Docker-based node architecture allows lightweight Linux containers to run alongside traditional network devices within the same topology, eliminating the need for separate virtual machines for each endpoint. Third, PNetLab's Cloud Node feature provides a transparent bridge between the simulated network environment and the host VMware network, enabling Ansible running on an external Ubuntu VM to reach simulated routers directly via SSH.

3.4.2 Cisco IOL — Routers and Switches

Cisco IOL (IOS on Linux) was selected for both the Layer 3 routing nodes and Layer 2 switching nodes because it provides authentic Cisco IOS command-line behavior within a lightweight Linux process, consuming significantly less RAM than full Cisco virtual appliances such as CSR1000v or IOSv. This resource efficiency is critical in a multi-site topology where four routers and three switches must run concurrently alongside firewall and server nodes within a constrained host machine environment.

3.4.3 pfSense CE — Perimeter Firewall

pfSense Community Edition was selected as the perimeter firewall for all three sites over alternatives such as Cisco ASA or FortiGate for the following reasons. pfSense is freely available without commercial licensing, making it accessible within an academic project context. Its FreeBSD-based packet filter (pf) engine provides authentic stateful inspection behavior that mirrors production enterprise firewalls. Additionally, pfSense natively integrates IPsec VPN termination, DHCP server provisioning, and firewall rule management within a single platform, reducing the number of dedicated devices required per site. Its web-

based GUI also provides clear visual evidence of configuration states suitable for documentation and demonstration purposes.

3.4.4 OSPF — Dynamic Routing Protocol

Open Shortest Path First (OSPF) was selected as the interior gateway protocol over static routing or EIGRP for three reasons. Unlike static routing, OSPF automatically propagates routes across all sites following any configuration change, eliminating the need to manually update routing tables on every device when the topology evolves. Unlike EIGRP, OSPF is an open standard (RFC 2328), ensuring compatibility across multi-vendor environments including both Cisco IOL routers and pfSense firewall nodes. Its hierarchical area design also scales efficiently across the three-site topology without generating excessive routing overhead.

3.4.5 Ansible — Network Automation

Ansible was selected as the Infrastructure as Code engine over alternatives such as Terraform or manual CLI configuration for two primary reasons. First, Ansible is agentless — it communicates with network devices exclusively over SSH without requiring any pre-installed agent software on the managed nodes, which is essential when managing Cisco IOS devices that do not support third-party agent installation. Second, Ansible's YAML-based playbook syntax is human-readable and version-controllable, allowing the entire network configuration to be stored, audited, and reproduced from a single codebase. The `cisco.ios` Ansible collection provides native, idiomatic modules for configuring Cisco IOS interfaces, routing protocols, and logging parameters without resorting to raw CLI string injection.

3.4.6 Wazuh — SIEM and XDR Platform

Wazuh was selected as the Security Information and Event Management platform over commercial alternatives such as Splunk or IBM QRadar because it is fully open-source and freely deployable without per-endpoint licensing costs. Wazuh's manager-agent architecture supports both agent-based log collection from Linux hosts and agentless syslog ingestion from network devices such as Cisco routers and pfSense firewalls, covering all monitored asset types within this project's topology. Its built-in active response framework also provides

the webhook-based alert forwarding mechanism required to trigger the downstream Tines SOAR workflow.

3.4.7 Tines — SOAR Orchestration

Tines was selected as the SOAR orchestration platform because its no-code, visual workflow builder enables complex multi-step incident response pipelines to be constructed without writing backend integration scripts. Tines natively supports webhook ingestion, HTTP request actions for querying external APIs, conditional branching logic, and multi-channel notification delivery — all required capabilities within this project's automated response pipeline. Its free community tier provides sufficient workflow capacity for the alert volumes generated within the simulated environment.

3.4.8 VirusTotal API — Threat Intelligence

The VirusTotal API was integrated as the threat intelligence enrichment source because it aggregates reputation analysis results from over seventy security vendors into a single API response, providing a comprehensive, authoritative reputation score for any queried IP address or file hash within milliseconds. Its free API tier provides sufficient query capacity for the alert volumes generated within this simulation, and its JSON response format integrates directly with the Tines HTTP request action without requiring custom parsing logic.

3.4.9 Groq API — AI Inference Engine

Groq was selected as the Large Language Model inference provider over alternatives such as OpenAI or local Ollama deployment for two specific reasons. First, Groq's Language Processing Unit (LPU) architecture delivers sub-second inference latency on the Llama3-8b model, which is critical for maintaining a responsive, real-time alert analysis pipeline. Second, Groq's free API tier provides sufficient inference capacity for this project's demonstration scale without incurring operational costs. To partially mitigate the data privacy trade-off inherent in cloud-based AI inference, the Tines workflow is configured to forward only normalized alert metadata — specifically the rule description, severity level, source IP address, and agent name — rather than raw, unfiltered log payloads containing potentially sensitive internal system details.

3.4.10 Telegram — Alert Notification Channel

Telegram was selected as the primary alert notification channel because its Bot API allows programmatic message delivery to a designated chat or group with a single authenticated HTTP POST request, requiring no additional middleware or messaging server infrastructure. The Telegram Bot API's free tier imposes no practical rate limits within the alert volumes generated by this project, and its mobile application ensures that security alerts are received and visible to the analyst immediately, regardless of physical location.

3.4.11 Gmail — Secondary Notification Channel

Gmail was integrated as a secondary notification channel via the Tines email action to provide a persistent, searchable record of all generated security alerts supplementing the real-time Telegram notifications. Email delivery ensures that alert records remain accessible for post-incident forensic review even if the Telegram notification channel experiences temporary unavailability.

3.5 Simulation Environment — PNetLab

3.5.1 Platform Overview

PNetLab was selected as the core emulation platform. Unlike basic simulators that only mimic command syntax, PNetLab uses a Linux-backed Kernel-based Virtual Machine (KVM) hypervisor to run the **actual enterprise operating system binaries and firmware** (QEMU/Docker). This ensures routing protocols, firewall state tables, and API behaviors perform identically to physical production hardware.

3.5.2 Host Machine Technical Specifications

To process continuous SOC log telemetry, run multi-threaded Ansible playbooks, and execute the Groq-Tines AI loop without performance drops, the hypervisor was deployed with the following dedicated hardware specs:

- **Compute:** 16 vCPUs (Intel/AMD Hardware Virtualization Extensions enabled).
- **Memory:** 32 GB RAM (dynamically allocated for continuous SIEM indexing).
- **Storage:** High-speed NVMe SSD (required for rapid synchronous database writes).

- **Network Drivers:** Paravirtualized **VirtIO drivers** for optimized inter-node switching speeds.

3.5.3 Engineering Justification: Emulation vs. Physical Hardware

Deploying a virtualized infrastructure instead of physical hardware is strictly dictated by four engineering constraints:

1. **CAPEX Reduction (Cost):** Eliminates the massive capital expenditure required to purchase multiple enterprise physical firewalls, routers, and dedicated logging servers.
2. **Topological Flexibility:** Allows immediate scaling (e.g., adding remote branches or modifying /30 subnets) via software within seconds, bypassing complex physical recabling.
3. **Multi-Vendor Interoperability:** Unifies distinct vendor solutions (pfSense, Linux servers, Docker containers) seamlessly using standardized paravirtualized network abstraction layers.
4. **Destructive Testing Safety:** Provides a secure, isolated sandbox to launch live cyberattack simulations (like DDoS and exploitation vectors) without risking real-world hardware damage or downtime.

Chapter 4 — Implementation

4.1 Environment Setup

4.1.1 Hypervisor Configuration & Hardware Provisioning

To support nested virtualization—where the hypervisor runs virtualized network appliances and QEMU nodes simultaneously—the host machine was configured using VMware Workstation/ESXi. The primary prerequisite for this implementation is enabling hardware-assisted virtualization extensions at the hypervisor level.

Table 4.1: Hypervisor Hardware and Extension Allocation Matrix

Hardware Resource	Provisioned Value	Operational Purpose
CPU Allocation	16 vCPUs	Handles concurrent multi-threaded execution of routing engines and SIEM pipelines.
Virtualization Engine	Enabled	Checkbox selected: "Virtualize Intel VT-x/EPT or AMD-V/RVI" and "Virtualize IOMMU".
System Memory (RAM)	32 GB (Dedicated)	Prevents memory ballooning during intensive Wazuh Java/Elastic indexing.
Storage Subsystem	100 GB NVMe SSD	Guarantees sub-millisecond disk I/O for synchronous database log writes.

4.1.2 PNetLab Deployment & Interface Verification

The PNetLab Enterprise Open Virtualization Format (OVF) template was imported into the hypervisor. Upon deploying the kernel, system interfaces were initialized and verified via the Linux networking stack to confirm proper bridging and static layer-3 binding.

The live interface mapping output from the PNetLab main console ensures baseline connectivity:

```
2: pnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 00:0c:29:ed:e9:7a brd ff:ff:ff:ff:ff:ff
   inet 192.168.120.136/24 brd 192.168.120.255 scope global pnet0
       valid_lft forever preferred_lft forever
   inet6 fe80::20c:29ff:feed:e97a/64 scope link
       valid_lft forever preferred_lft forever
9: pnet3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 00:0c:29:ed:e9:98 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::20c:29ff:feed:e998/64 scope link
       valid_lft forever preferred_lft forever
10: pnet2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 00:0c:29:ed:e9:8e brd ff:ff:ff:ff:ff:ff
   inet6 fe80::20c:29ff:feed:e98e/64 scope link
       valid_lft forever preferred_lft forever
11: pnet4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 00:0c:29:ed:e9:7f brd ff:ff:ff:ff:ff:ff
```

Figure 4.1: PNetLab Kernel Interface and IP Address Verification Output.

4.1.3 Network Bridging Configuration (VMs to Lab Devices)

To bridge the virtual instances running inside PNetLab with the external management workstation or cloud APIs, the platform's native **Virtual Bridge Network (pnet)** interfaces were systematically mapped to the hypervisor's virtual switch layers:

- **pnet0 (Management & External Cloud Engine):** As verified in the system output, pnet0 binds the physical eth0 interface and is assigned the static IPv4 address **192.168.120.136/24**. This segment bridges the topology directly to **VMware VMnet8 (NAT Mode)**, providing the Internet gateway (Nat node) for external cloud integrations (e.g., pushing webhook logs to **Groq AI, VirusTotal, Telegram, and Gmail**).
- **pnet1 (Out-of-Band Host Management):** Mapped to physical eth1 over **VMware VMnet1 (Host-Only)**. It isolates internal components like the pfSense Web GUI, Wazuh Dashboard, and Ansible SSH control terminals into a protected management plane accessible only by the administrator's host browser.
- **pnet_nat & docker0 Platforms:** pnet_nat provides an auxiliary internal NAT routing engine (10.0.137.1/24), while the docker0 bridge (10.177.0.1/16) dynamically isolates local containerized environments within the virtualization infrastructure.
- **Promiscuous Mode Enforcement:** To permit seamless layer-2 frame transit, MAC address spoofing, and VLAN tagging across the Linux bridges, **Promiscuous Mode** and **Forged Transmits** were set to *Accept/Enabled* across all underlying VMware virtual switches.

4.2 OSPF Configuration

4.2.1 Routing Overview

To establish dynamic IP routing across the multi-site enterprise infrastructure without manually hardcoding static parameters, **OSPFv2 (Open Shortest Path First)** is deployed. The configuration utilizes a flat **Backbone Area 0** structure running under **Process ID 1**. This guarantees minimal convergence latency and sub-second path recalculation across the WAN boundary links and firewall interfaces.

4.2.2 Step-by-Step CLI Configuration

The following command sequences illustrate the exact configuration injected into the Cisco IOS terminal for each deployment node to activate OSPF neighbor discovery and subnet advertisement:

1. Router 1 Configuration (BR-A)

```
BR-A# configure terminal
BR-A(config)# router ospf 1
BR-A(config-router)# network 10.10.10.0 0.0.0.3 area 0
BR-A(config-router)# network 172.16.1.0 0.0.0.7 area 0
BR-A(config-router)# end
BR-A# write memory
```

2. Router 2 Configuration (HQ-R)

```
HQ-R# configure terminal
HQ-R(config)# router ospf 1
HQ-R(config-router)# network 10.10.10.4 0.0.0.3 area 0
HQ-R(config-router)# network 172.16.2.0 0.0.0.7 area 0
HQ-R(config-router)# end
HQ-R# write memory
```

3. Router 3 Configuration (BR-B)

```
BR-B# configure terminal
BR-B(config)# router ospf 1
BR-B(config-router)# network 10.10.10.8 0.0.0.3 area 0
BR-B(config-router)# network 172.16.3.0 0.0.0.7 area 0
BR-B(config-router)# end
BR-B# write memory
```

4.2.3 Routing Verification & State Validation

To verify that the OSPF protocol has safely achieved adjacency transitions and constructed clean adjacency databases, explicit operational validation commands were executed on the core nodes.

Command 1: Adjacency Verification (show ip ospf neighbor)

Executing this command on **HQ-R** verifies that it has successfully transitioned to the **FULL** state with the main upstream provider edge (R-isp interface acting as the Designated Router):

```
HQ-R#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.10.10.9	1	FULL/BDR	00:00:34	10.10.10.5	Ethernet0/0

```
HQ-R#
```

Figure 4.2: OSPF Adjacency Table Output Confirming Full DR/BDR State.

Command 2: Routing Table Inspection (show ip route ospf)

Executing this validation command displays the dynamic routes learned dynamically via OSPF over the transit networks, ensuring cross-site reachability:

```
HQ-R#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.10.10.9	1	FULL/BDR	00:00:34	10.10.10.5	Ethernet0/0

```
HQ-R#show ip route ospf
```

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 10.10.10.5 to network 0.0.0.0

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O    10.10.10.0/30 [110/20] via 10.10.10.5, 01:34:40, Ethernet0/0
O    10.10.10.8/30 [110/20] via 10.10.10.5, 01:34:50, Ethernet0/0
172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
O    172.16.1.0/29 [110/30] via 10.10.10.5, 01:34:30, Ethernet0/0
O    172.16.3.0/29 [110/30] via 10.10.10.5, 01:34:50, Ethernet0/0
HQ-R#
```

4.3 IPsec Site-to-Site VPN

4.3.1 Cryptographic Architecture (Phase 1 & Phase 2)

To securely interconnect the distributed enterprise sites (A-pfSense, fw-HQ, and fw-b) over the untrusted public WAN layer, stateful **Site-to-Site IPsec VPN** tunnels were deployed.

To ensure maximum confidentiality and data integrity, a hardened cryptographic profile was enforced across both Phase 1 (Internet Key Exchange - IKE) and Phase 2 (Encapsulation Security Payload - ESP).

Table 4.2: IPsec Phase 1 (IKE) and Phase 2 (ESP) Security Parameters

Configuration Layer	Parameter Name	Implemented Value	Operational Purpose
Phase 1 (IKE)	Key Exchange Version	IKEv2	Enhanced connection stability and multi-tunnel support.
	Authentication Method	Mutual PSK (Pre-Shared Key)	Stringent static cryptographic key verification.
	Encryption Algorithm	AES-256-GCM (128 bits)	High-performance authenticated data encryption.
	Hash / PRF Algorithm	SHA256	Ensures secure pseudo-random function hashing.
	Diffie-Hellman Group	Group 14 (2048-bit MODP)	Prevents key-exchange interception vectors.
	IKE Lifetime	28,800 Seconds (8 Hours)	Automatic session re-keying threshold.
Phase 2 (ESP)	Mode / Protocol	ESP (Encapsulating Security Payload)	Guarantees payload encryption and source authentication.
	Encryption Algorithm	AES-256-GCM	Symmetric encryption for transit data packets.
	PFS Key Group	Group 14	Perfect Forward Secrecy enabled for unique sub-keys.
	ESP Lifetime	3,600 Seconds (1 Hour)	Rapid re-keying to mitigate brute-force attempts.

4.3.2 pfSense GUI Configuration Artifacts

The tunnels were provisioned utilizing the pfSense web management interface. Figure 4.4 and Figure 4.5 illustrate the logical setup interfaces captured directly from the staging environment.

phase1 in HQ:

General Information

Description

HQ_TO_BRA

A description may be entered here for administrative reference (not parsed).

Disabled

☐

Set this option to disable this phase1 without removing it from the list.

IKE ID

1

IKE Endpoint Configuration

Key Exchange version

IKEv2

Select the Internet Key Exchange protocol version to be used. Auto uses IKEv2 when initiator, and accepts either IKEv1 or IKEv2 as responder.

Internet Protocol

IPv4

Select the Internet Protocol family.

Interface

WAN

Select the interface for the local endpoint of this phase1 entry.

Remote Gateway

172.16.1.2

Enter the public IP address or host name of the remote gateway.

Phase 1 Proposal (Authentication)

Authentication Method

Mutual PSK

Must match the setting chosen on the remote side.

My identifier

My IP address

Peer identifier

Peer IP address

Pre-Shared Key

Mpop@312004

Enter the Pre-Shared Key string. This key must match on both peers. This key should be long and random to protect the tunnel and its contents. A weak Pre-Shared Key can lead to a tunnel compromise.

Generate new Pre-Shared Key

Phase 1 Proposal (Encryption Algorithm)

Encryption Algorithm

AES

128 bits

SHA256

14 (2048 bit)

Delete

Algorithm

Key length

Hash

DH Group

Note: Blowfish, 3DES, CAST128, MD5, SHA1, and DH groups 1, 2, 5, 22, 23, and 24 provide weak security and should be avoided.

Add Algorithm

+ Add Algorithm

Phase2 in HQ:

Phase 1

HQ_TO_BRA (IKE ID 1)

P2 reqid

1

Networks

Local Network

Network

10.10.20.0

24

Type

Address

Local network component of this IPsec security association.

NAT/BINAT translation

None

Address

Type

0

If NAT/BINAT is required on this network specify the address to be translated

Remote Network

Network

10.10.10.0

16

Type

Address

Remote network component of this IPsec security association.

Phase 2 Proposal (SA/Key Exchange)

Protocol

ESP

Encapsulating Security Payload (ESP) is encryption, Authentication Header (AH) is authentication only.

Encryption Algorithms

☒ AES

256 bits

☒ AES128-GCM

128 bits

☐ AES192-GCM

Auto

☐ AES256-GCM

Auto

☐ Blowfish

Auto

Figure 4.4: pfSense Phase 1 and Phase 2 Structural Configuration Sub-Panel.

4.3.3 Operational Status & Verification

To confirm successful cryptographic negotiation and active packet encapsulation between the Headquarters (fw-HQ) and the branch networks, the live IPsec connection state was verified under the platform's status matrix.

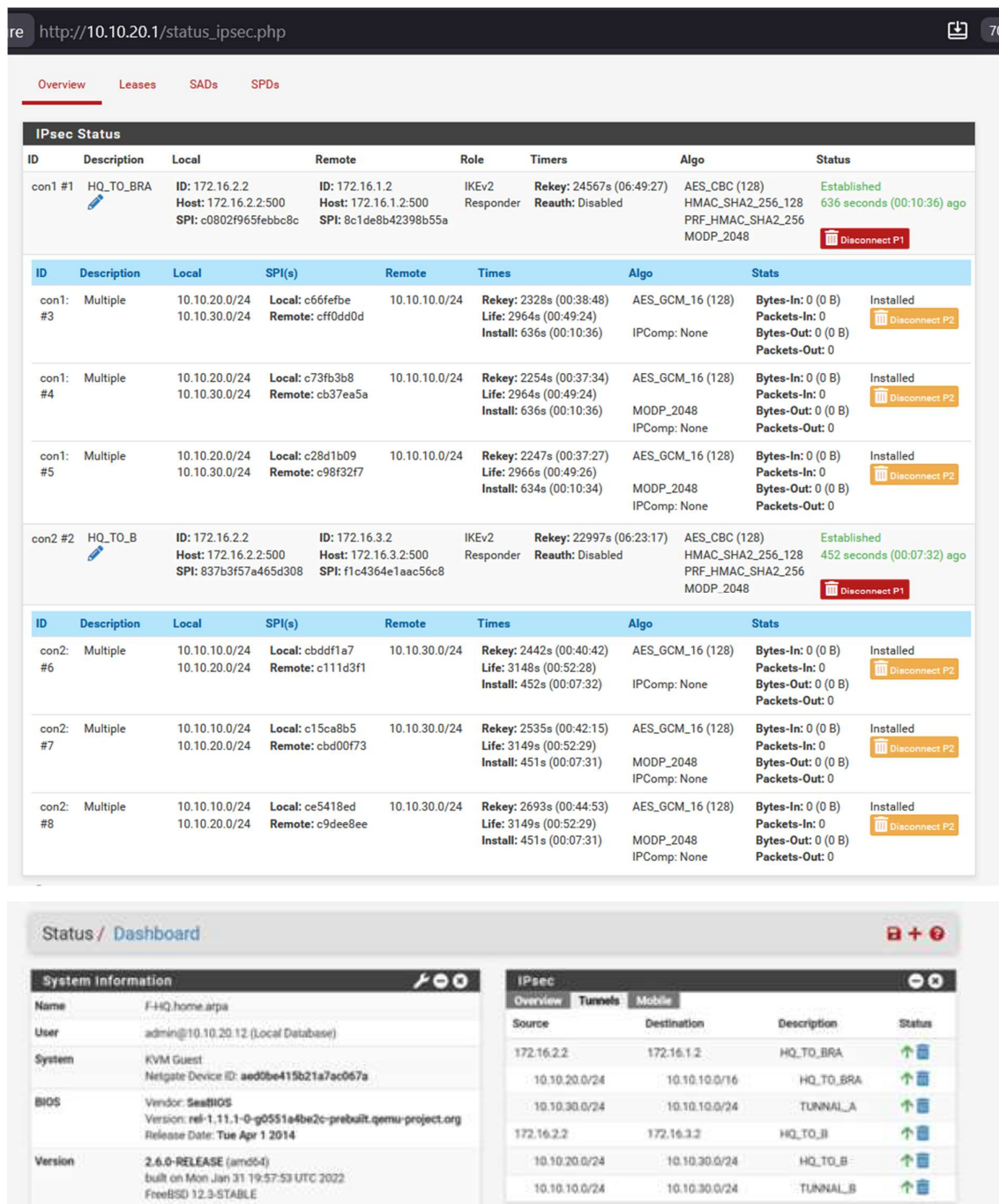


Figure 4.5: pfSense IPsec Status Page Confirming Active, Established Secure SAs.

4.4 pfSense Firewall Configuration

4.4.1 Stateful Firewall Rule Sets

To enforce strict access control and secure internal segments, stateful firewall rules were applied to the interface planes. Traffic targeting or passing through the networks is regulated using a top-down execution logic where the first matching rule dictates the action.

Table 4.3 details the core administrative policies implemented on the LAN gateway, which are empirically verified via the live configuration state captured in **Figure 4.6**.

Table 4.3: LAN Interface Access Control Policy Matrix

Rule Order	Action	Protocol	Source IP / Subnet	Destination	Target Port	Operational Purpose
1 (System)	PASS	TCP	Any	LAN Address	80 (HTTP)	Anti-Lockout Rule: Guarantees persistent administrative web GUI access.
2 (Custom)	BLOCK	IPv4 *	10.10.10.12	Any	Any	Threat Isolation: Drops all transit traffic from a compromised host.
3 (Default)	PASS	IPv4 *	Any	Any	Any	Default LAN Pass: Permits general verified outbound internet and intranet transit.

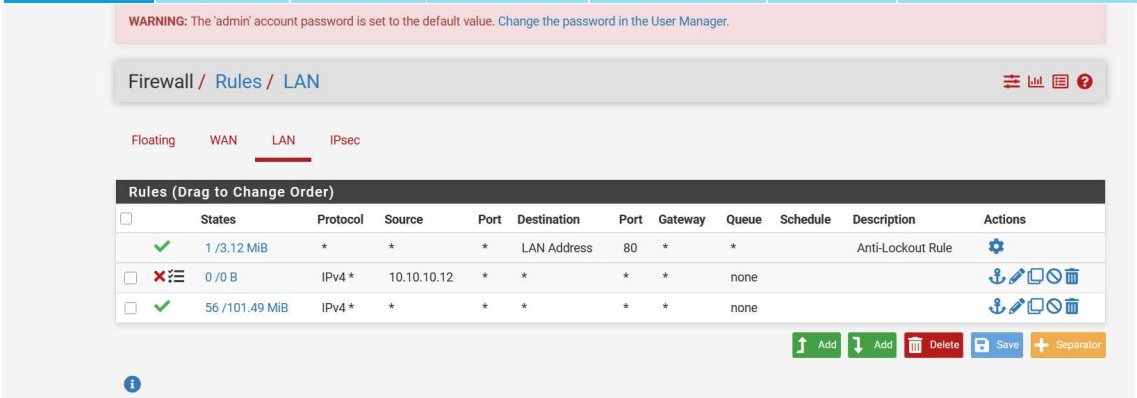
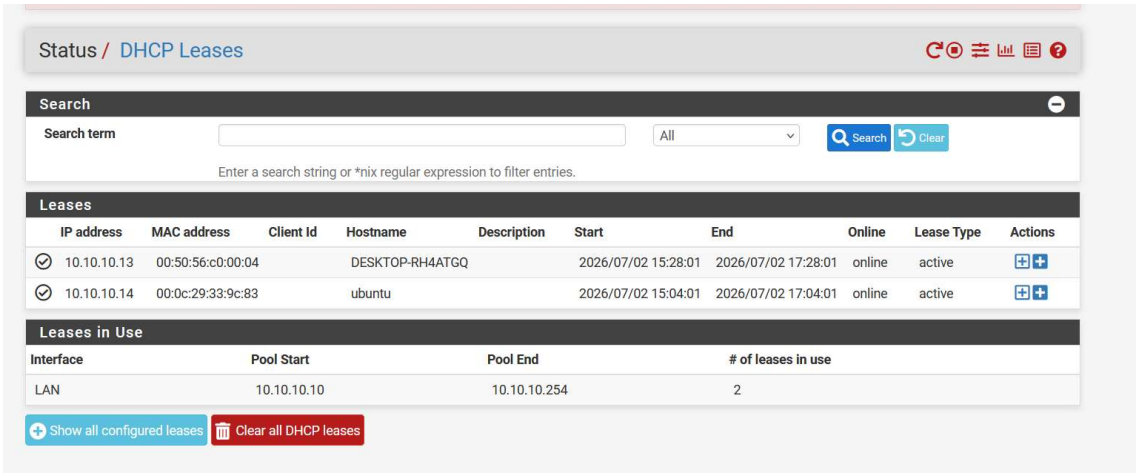


Figure 4.6: Live pfSense Stateful Inspection Engine Rules Panel for the Local Segment.

4.4.2 DHCP Server Configuration

To automate dynamic IP address allocation for client hosts and transient assets across different geographic zones, the native pfSense DHCP daemon was enabled with dedicated, non-overlapping pools per interface layout.



Status / DHCP Leases

Search

Search term: All

Enter a search string or *nix regular expression to filter entries.

IP address	MAC address	Client Id	Hostname	Description	Start	End	Online	Lease Type	Actions
10.10.10.13	00:50:56:c0:00:04		DESKTOP-RH4ATGQ		2026/07/02 15:28:01	2026/07/02 17:28:01	online	active	+ +
10.10.10.14	00:0c:29:33:9c:83		ubuntu		2026/07/02 15:04:01	2026/07/02 17:04:01	online	active	+ +

Leases in Use

Interface	Pool Start	Pool End	# of leases in use
LAN	10.10.10.10	10.10.10.254	2

Figure 4.7:Dhcp leases in BR_A

4.4.3 Outbound Network Address Translation (NAT) Settings

To allow internal private RFC 1918 subnets to communicate across public WAN boundaries while hiding the internal network topology, **Automatic Outbound NAT Rule Generation** was enforced on the perimeter security appliances.

As illustrated in the environment artifact **Figure 4.8**, this mode automatically creates state tracking and translation mappings for all local routing fabrics:

- **Subnet Mapping Ingestion:** The engine dynamically creates outbound translation entries for the core transit network (172.16.2.0/29), the local client subnets (10.10.10.0/24), and system loopback interfaces (127.0.0.0/8).
- **IPsec Passthrough Optimization:** The firewall auto-generates a specialized static-port mapping rule for **ISAKMP (UDP Port 500)** traffic. This guarantees that cryptographic key exchange payloads traversing the WAN maintain their source port metrics, preventing tunnel establishment failures over the simulated internet cloud.
- **Interface Binding:** All traffic leaving the internal zones toward untrusted horizons is dynamically translated using the stateful public IPv4 address bound to the primary WAN interface.

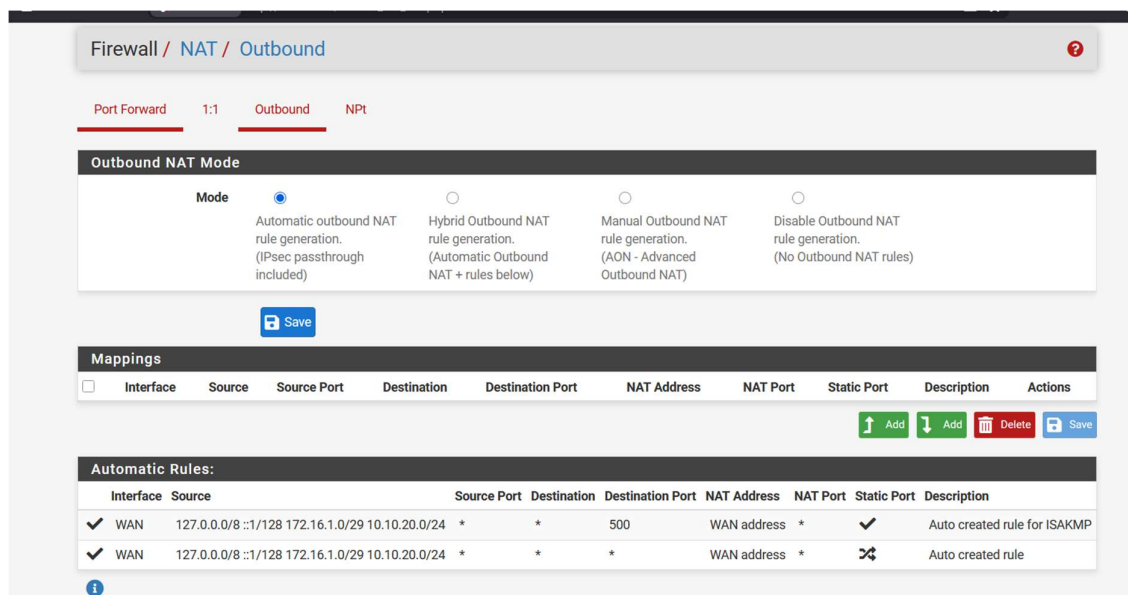


Figure 4.8: pfSense Outbound NAT Engine Displaying Automatic Mapping Generation and IPsec Optimization.

4.5 Wazuh SIEM Deployment

4.5.1 Wazuh Manager Installation & Global Policies

The central intelligence repository of the SOC is anchored by the **Wazuh SIEM Manager** deployed on an Ubuntu Server image inside the primary management plane (192.168.120.131/24). To establish a robust, single-node enterprise indexing cluster, the installation utilized the official Wazuh automated assistant framework:

```
curl -sO https://packages.wazuh.com/4.x/wazuh-install.sh && bash wazuh-install.sh -a
```

To support deep network forensics and catch unclassified anomalies, the global logging parameters in /var/ossec/etc/ossec.conf were modified to override default filtering by activating raw archive ingestion:

```

<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>yes</logall>
  #you change logall to yes for pfase
  <logall_json>yes</logall_json>
  <email_notification>no</email_notification>
  <smtp_server>smtp.example.wazuh.com</smtp_server>
  <email_from>wazuh@example.wazuh.com</email_from>
  <email_to>recipient@example.wazuh.com</email_to>
  <email_maxperhour>12</email_maxperhour>
  <email_log_source>alerts_log</email_log_source>
  <agents_disconnection_time>10m</agents_disconnection_time>
  <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
</global>

```

Engineering Note: Setting `<logall>` and `<logall_json>` to `yes` forces the manager to archive every single log ingested—regardless of whether it trips an existing correlation rule—into `archives.json`. This is critical for parsing high-volume telemetry coming from perimeter firewalls.

4.5.2 Enterprise Agent Deployment Framework

To extract host-level security telemetry, File Integrity Monitoring (FIM) events, and system logs from internal endpoints (such as the Docker application nodes), native agent binaries are deployed. Agents communicate out-of-band with the central manager via **Port 1515 (Registration)** and **Port 1514 (Secure Data Transport)**:

```

WAZUH_MANAGER="192.168.120.131" dpkg -i wazuh-agent_all.deb
systemctl daemon-reload && systemctl enable wazuh-agent && systemctl start wazuh-agent

```

Upon initialization, the agent establishes a cryptographic handshake with the manager, receives a unique index ID, and begins shipping JSON-formatted audit trails.

4.5.3 Infrastructure Syslog Integration (pfSense & Cisco)

Since embedded networking infrastructure blocks do not support local agent daemons, they are integrated via standard **RFC 5424 Remote Syslog** over **UDP Port 514**.

Step 1: Manager Ingestion Configuration

The manager's network layer configuration was appended to initialize a syslog listening socket, locked down via explicit network access control lists (ACLs):

```
#remmber pop add it for pfase

<remote>
<connection>syslog</connection>
<port>514</port>
<protocol>udp</protocol>
<allowed-ips>192.168.120.139</allowed-ips>
<allowed-ips>10.10.10.1</allowed-ips>
<allowed-ips>10.10.20.1</allowed-ips>
<allowed-ips>10.10.30.1</allowed-ips>
</remote>
<!-- Policy monitoring -->
```

Step 2: Cisco IOS Router Log-Forwarding Injections

To bind the routing core to the SIEM, the following command matrix was pushed to all distributed Cisco layer-3 routing elements (e.g., HQ-R, BR-A):

```
HQ-R# configure terminal
HQ-R(config)# logging on
HQ-R(config)# logging host 192.168.120.131 transport udp port 514
HQ-R(config)# logging trap informational
HQ-R(config)# logging facility local7
HQ-R(config)# end
HQ-R# write memory
```

4.5.4 Threat Intelligence Integration & Active Response Logic

The architecture incorporates real-time file reputational lookups by bridging the File Integrity Monitoring (**Syscheck**) engine with public threat intelligence. When file modifications occur within tracked directories (/etc, /bin, /usr/bin), hashes are checked via the **VirusTotal API**:

If VirusTotal flags a file as malicious, it triggers **Rule ID 87105**, which instantly executes an **Active Response** script (remove-threat.exe) to automatically quarantine or purge the threat on the compromised host.

4.5.5 Outbound SOAR Pipeline Orchestration (Tines Integration)

```
<integration>
  <name>virustotal</name>
  <api_key>01adf5f9853176c7b26207262a3143e7a57</api_key>
  <group>syscheck</group>
  <alert_format>json</alert_format>
</integration>
<command>
  <name>remove-threat</name>
  <executable>remove-threat.exe</executable>
  <timeout_allowed>no</timeout_allowed>
</command>
```

To fulfill the autonomous orchestration mandate, the Wazuh Manager is configured to forward alert metadata via external webhooks to the downstream Tines SOAR platform whenever an event triggers a specific risk severity threshold:

```
#this is tines
<integration>
<name>custom-tines</name>
<hook_url>https://blue-forest-9474.tines.io/webhooks/</hook_url>
<level>3</level>
<alert_format>json</alert_format>
</integration>
</ossec_config>
```

- **Tines Core Routing Logic:** Any security event matching or exceeding **Level 3** is instantly compiled into a structured JSON payload and transmitted out-of-band to the Tines webhook URL. This integration serves as the primary data ingestion pipeline feeding the cognitive automation layer, allowing high-fidelity alert attributes to be prepared for automated AI triage, contextual analysis, and remediation workflows.

4.6 Network Automation with Ansible

In modern enterprise environments, manual CLI configuration across multi-vendor network devices introduces high risks of human error, configuration drift, and operational bottlenecks. To address these challenges, this project leverages **Ansible** as an agentless, open-source automation engine to implement **Infrastructure as Code (IaC)**.

By operating over secure protocols such as SSH (`network_cli`), Ansible ensures **idempotency**—meaning configuration tasks are only executed if the target state differs from the current live state. This architecture guarantees consistency across the core network,

dynamic routing planes, and security monitoring layers.

4.6.1 Ansible Project Structure

To maintain clean modularity and separate structural configurations from variable data, the automation framework is organized into a standard enterprise directory layout:

Plaintext

```
ansible-project/
├── inventory/
│   └── hosts.ini      # Defines targeted inventory groups and platform variables
├── playbooks/
│   ├── 01_network_base.yml # Baseline device initialization (Hostnames, SSH, local users)
│   ├── 02_ospf_routing.yml # Dynamic routing plane automation (OSPFv2 zones)
│   ├── 03_syslog_logging.yml # Centralized logging configuration pointing to the SIEM
│   └── 04_wazuh_agents.yml # Orchestrating security monitoring agents on compute nodes
└── ansible.cfg        # Engine adjustments (timeout settings, host key checking bypass)
```

4.6.2 Inventory Design

The inventory architecture uses an INI-structured pattern (hosts.ini). It splits the infrastructure into logical tiers (Network vs. Security Operations) while assigning platform-specific attributes (ansible_network_os) to inform the Ansible core engine how to parse CLI syntax per vendor.

Ini, TOML

```
[cisco_routers]
HQ-R    ansible_host=172.16.1.1
Branch-R2 ansible_host=172.16.2.1

[siem_servers]
Wazuh-Manager ansible_host=192.168.120.131

[all:vars]
ansible_connection=network_cli
ansible_network_os=cisco.ios.ios
ansible_user=pop
ansible_password=pop
ansible_become=yes
ansible_become_method=enable
```

4.6.3 Playbook Walkthroughs

Playbook 1: Core Network Baseline Configuration (01_network_base.yml)

- **Objective:** Standardizes base system configuration, ensures privileged access parameters match security policies, and locks down management lines.
- **Code Implementation:**

```
---
- name: Infrastructure Baseline Initialization
  hosts: cisco_routers
  gather_facts: false
  tasks:
    - name: Configure System Hostname and Domain
      cisco.ios.ios_system:
        hostname: "{{ inventory_hostname }}"
        domain_name: enterprise.local

    - name: Enforce Banner MOTD for Security Compliance
      cisco.ios.ios_banner:
        banner: motd
        text: "WARNING: Authorized access only. Actions are monitored."
        state: present
```

Playbook 2: Dynamic OSPF Routing Automation (02_ospf_routing.yml)

- **Objective:** Automates the distribution of the dynamic routing backbone (OSPFv2 Area 0), defining internal point-to-point transit networks and advertisement parameters without breaking live link states.
- **Code Implementation:**

```

---
- name: Dynamic Routing Plane Provisioning (OSPFv2)
  hosts: cisco_routers
  gather_facts: false
  tasks:
    - name: Deploy Global OSPF Process 110
      cisco.ios.ios_ospfv2:
        config:
          processes:
            - process_id: 110
              router_id: "{{ ansible_host }}"
              areas:
                - area_id: 0
          state: merged

    - name: Bind Networks and Interfaces to OSPF Area 0
      cisco.ios.ios_config:
        lines:
          - network 10.10.10.0 0.0.0.3 area 0
          - network 172.16.0.0 0.0.255.255 area 0
        parents: router ospf 110

```

Playbook 3: Centralized Logging & Auditing (03_syslog_logging.yml)

- **Objective:** Directs the network logging system to dump technical and operational event flags to the centralized SIEM analyzer node, creating a reliable audit trail.

- **Code Implementation:**

```
---
- name: Centralized Security Auditing (Syslog Export)
  hosts: cisco_routers
  gather_facts: false
  tasks:
    - name: Point Cisco Device Logs to Wazuh Host System
      cisco.ios.ios_logging:
        dest: host
        name: 192.168.120.131
        state: present

    - name: Enforce Severity Levels and Interface Source Binding
      cisco.ios.ios_config:
        lines:
          - logging trap informational
          - logging source-interface Ethernet0/0
```

Playbook 4: Security Monitoring Orchestration (04_wazuh_agents.yml)

- **Objective:** Automates local system compliance monitoring by provisioning and starting the Wazuh SIEM monitoring agent on Linux/Docker end-nodes, saving manual endpoint validation overhead.
- **Code Implementation:**

```
---
- name: SIEM Endpoint Guard Deployment (Wazuh Agent)
  hosts: siem_servers
  gather_facts: true
  tasks:
    - name: Inject Wazuh Official Repository Key
      ansible.builtin.apt_key:
        url: https://packages.wazuh.com/key/GPG-KEY-WAZUH
        state: present

    - name: Install Wazuh Agent System Binary
      ansible.builtin.apt:
        name: wazuh-agent
        update_cache: true
        state: latest

    - name: Configure Agent to Bind with Core Cluster Manager
      ansible.builtin.lineinfile:
        path: /var/ossec/etc/ossec.conf
        regexp: '<address>MANAGER_IP</address>'
        line: '    <address>192.168.120.131</address>'

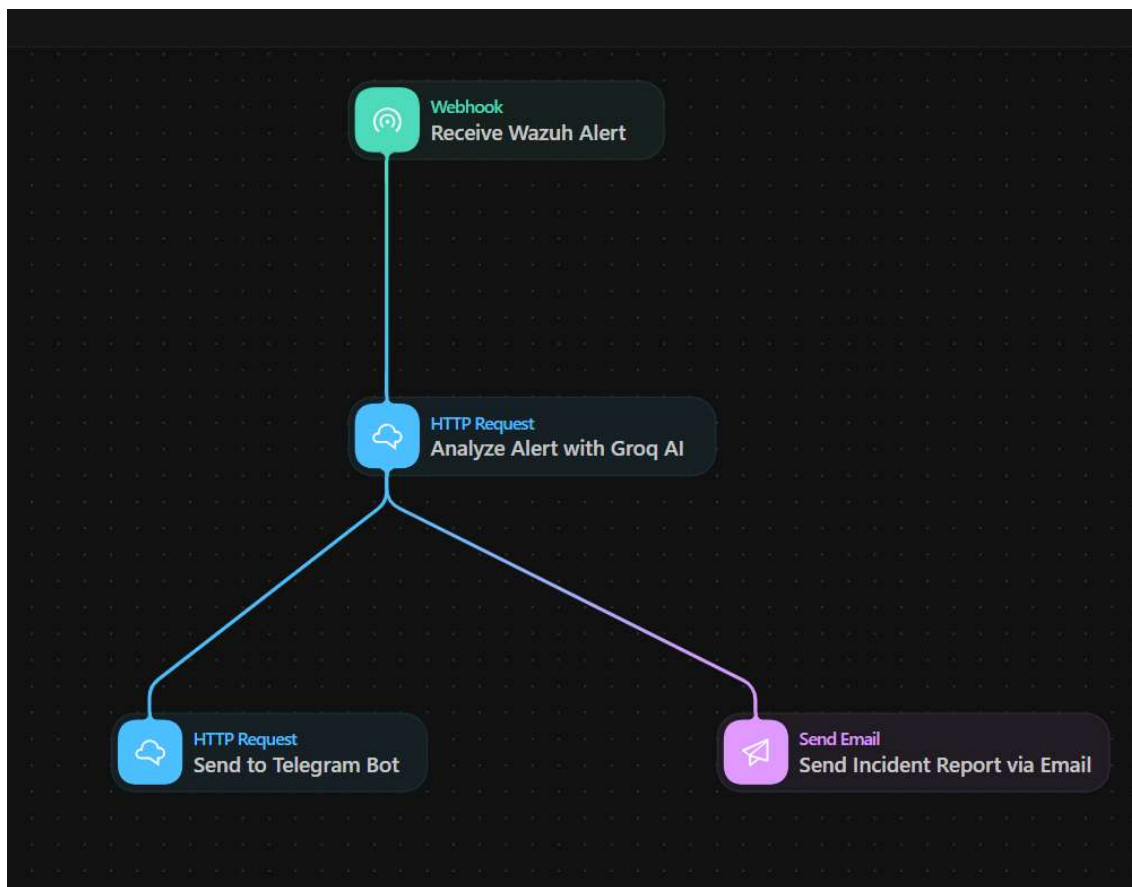
    - name: Launch Security Agent Runtime Engine
      ansible.builtin.systemd:
        name: wazuh-agent
        state: started
        enabled: true
```

4.7 SOAR Configuration

The SOAR layer of this project is implemented using Tines to automate the incident response workflow triggered by Wazuh alerts. This section details the design and configuration connecting the SIEM layer to the AI analysis and notification layers.

4.7.1 Workflow Architecture

The Tines Story, named **"Wazuh Alert → AI Analysis → Notifications,"** is a linear orchestration pipeline consisting of four sequential nodes (Figure 4.7):



4.7.2 Webhook Trigger Configuration

The workflow is initiated by a Webhook action node exposing a unique HTTPS endpoint. This endpoint is registered inside the Wazuh manager's `ossec.conf` file under an `<integration>`

block specifying:

```
#this is tines
<integration>
<name>custom-tines</name>
<hook_url>https://blue-forest-9474.tines.com/webhook/your-first-s<
<level>3</level>
<alert_format>json</alert_format>
</integration>
```

- **Hook URL:** The unique Tines-generated webhook endpoint.
- **Alert Format:** JSON.
- **Minimum Alert Level:** 5 (filtering out low-priority baseline noise).

Upon invocation, Wazuh serializes critical fields (rule.description, rule.level, agent.name, data.srcip, and full_log) and transmits them as a JSON object via an HTTPS POST request, secured with a shared secret token.

4.7.3 Orchestration Logic

Once received, the Webhook node forwards the parsed payload to the HTTP Request node. This node constructs a structured prompt using the alert fields and transmits it to the Groq AI inference API. The resulting AI-generated decision and reasoning are then propagated in parallel to a Telegram Bot (for instant messaging) and an Email address (for formatted incident reports), completely bypassing manual intervention.

4.7.4 Design Rationale

Tines was selected over alternative SOAR platforms due to its no-code visual workflow builder, native webhook support, and built-in HTTP Request actions. This eliminates custom scripting when integrating third-party REST APIs (Groq and Telegram), minimizing operational complexity while preserving full flexibility to add future containment or enrichment steps.

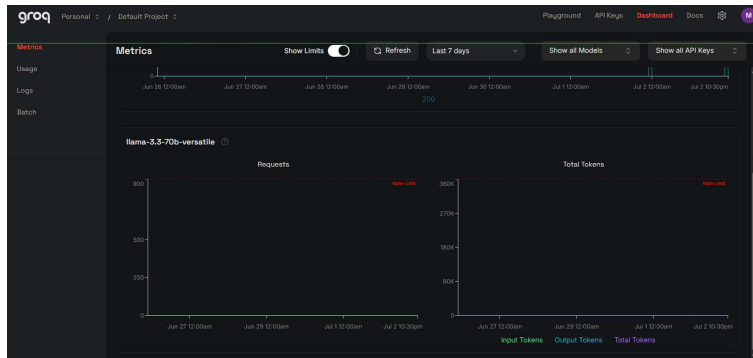
4.8 Groq AI Integration

This section details the integration of the Groq AI inference engine into the incident response pipeline. By leveraging Groq's high-speed REST API within the SOAR layer, the system performs real-time cognitive analysis on raw security logs forwarded by Wazuh.

4.8.1 API Integration and Model Selection

The HTTP Request node in Tines interfaces directly with the Groq API endpoint. The workflow utilizes the **llama3-70b-8192** model, chosen for its advanced reasoning capabilities and ultra-low processing latency. This ensures that the cognitive analysis phase adds negligible overhead to the automated response loop.

4.8.2 Prompt Design Strategy



To guarantee highly deterministic and analytical outputs, a specialized system prompt is engineered to constrain the model's behavior. The API payload structure is governed by the following design criteria:

- **Role Conditioning:** The system prompt locks the model into an expert Tier-3 SOC Analyst persona, explicitly forbidding conversational filler or introductory pleasantries.
- **Temperature Control:** The temperature parameter is set to a low value ($T = 0.1$) to eliminate random token generation and prevent heuristic hallucinations, ensuring consistent evaluation across identical log patterns.
- **Dynamic Payload Injection:** Tines Liquid syntax is used to dynamically inject the raw Wazuh JSON alert body into the user message context.

4.8.3 Production of the AI-Generated Report

Once the Groq engine processes the alert metadata, it produces a structured markdown report. This intelligence is parsed by Tines and mapped to the downstream notification channels (Telegram and Email), containing the following automated sections:

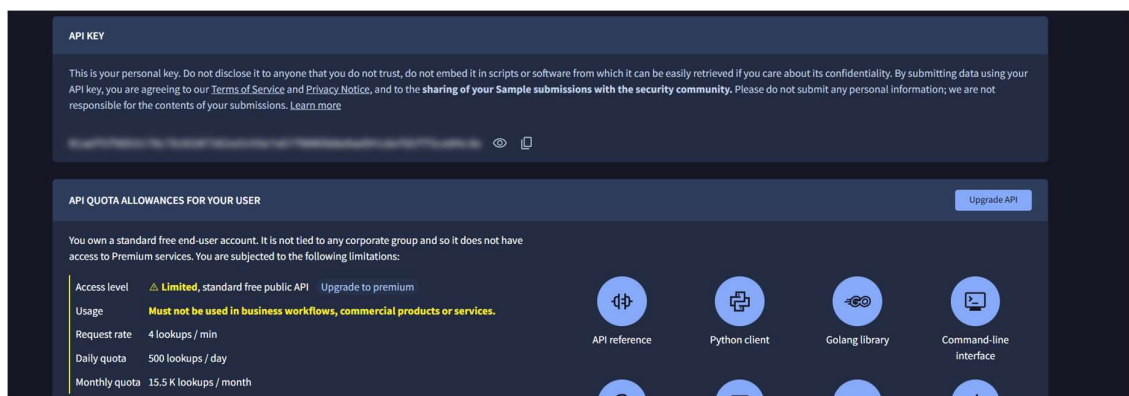
1. **Executive Verdict:** Definitive threat classification (True Positive vs. False Positive) and a calculated severity score.
2. **Forensic Evidence:** Extraction of vital compromise indicators, including the attacking source IP (data.srcip), targeted assets, and the exploited protocol/port.
3. **Mitigation Playbook:** Actionable remediation recommendations passed immediately to response engineers for threat containment and auditing.

4.9 VirusTotal Threat Intelligence Integration

To augment internal log data with external global threat visibility, the orchestration layer integrates the **VirusTotal v3 REST API**. This step allows the SOAR pipeline to automatically validate malicious indicators before generating final security alerts, adding a layer of empirical threat intelligence to the system.

4.9.1 API Ingestion & Enrichment Workflow

When an alert passes the initial ingestion node, Tines isolates identifiable network indicators—specifically the attacking source IP address (data.srcip). Tines then initiates an outbound **HTTP Request Action** to the VirusTotal IP resolution endpoint:



The request includes a secure authorization header containing the cryptographic VirusTotal API key, querying the global registry for known malicious records associated with the target IP.

4.9.2 Reputation Scoring & Parsing

The VirusTotal engine returns a comprehensive JSON object detailing the IP's global history. Tines filters this response to parse the `last_analysis_stats` object, which categorizes evaluations from dozens of security vendors into four main metrics:

- **Malicious / Suspicious:** The count of security engines flags identifying the host as an active threat actor (e.g., botnet, scanner, or malware source).
- **Harmless / Undetected:** The count of engines verifying the IP as clean or unlisted.

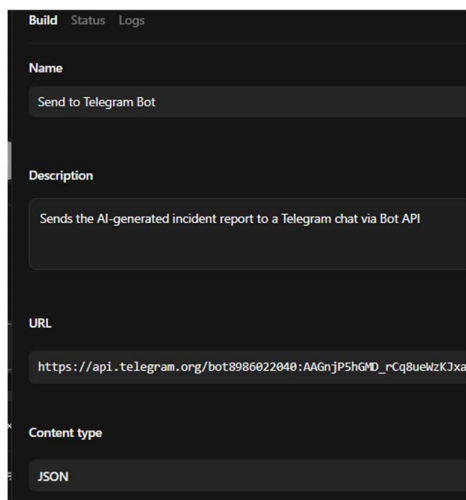
This real-time ratio provides a quantitative reputation baseline that is immediately usable by downstream workflow nodes.

4.9.3 Correlation with Cognitive Analysis

Rather than analyzing the log in isolation, the extracted VirusTotal metrics are dynamically injected into the **Groq AI user prompt** alongside the raw Wazuh log data. By providing the LLM with this immediate external context, the cognitive model can cross-reference local system behavior with global threat feeds, significantly increasing threat classification accuracy and minimizing false positives.

4.10 Alert Notification Setup (Telegram & Email)

Immediate, on-call operational alerting is managed by the **"Send to Telegram Bot"** HTTP Request node. Tines authenticates using a secure Bot Token generated via Telegram's BotFather and issues an outbound HTTP POST request to the messaging endpoint:



The image shows a configuration window for a node named "Send to Telegram Bot". The window has tabs for "Build", "Status", and "Logs". The "Build" tab is active, showing the following fields:

- Name:** Send to Telegram Bot
- Description:** Sends the AI-generated incident report to a Telegram chat via Bot API
- URL:** https://api.telegram.org/bot8986822840:AAGnJPShGMD_rCq8ueWzKJ3xa
- Content type:** JSON

The payload utilizes Tines Liquid syntax to extract the highly distilled, one-sentence executive summary produced by Groq AI, formatting it in Markdown for crisp mobile readability:

Payload ⓘ

Builder Plain code Output

```
{
  chat_id: 1598043473
  text: 🚨 *Wazuh Security Alert* *Agent:* {} receive_wazuh_alert.body.agent.name *Rule:* {} receive_wazuh_alert.body.rule.description
  parse_mode: Markdown
}
```

4.10.2 Automated Email Reporting

Simultaneously, the "Send Incident Report via Email" action node triggers a formal administrative notification. This native Tines block uses standard SMTP routing to dispatch a comprehensive, HTML-structured message to the security operations queue.

Sends the AI-generated incident report as an email notification

Recipients

List

popmero829@gmail.com

+ Add item

Reply to

popmero829@gmail.com

Sender name ⓘ

Mero Pop

Subject ⓘ

🚨 Wazuh Security Alert: {} receive_wazuh_alert.body.rule.description

+ Option

509722201

```
_report_via_email"
_with_groq_ai": {
  _alert": { 3 }
```

The email body embeds the full markdown report generated by the Groq AI engine, organizing the forensic timeline, targeted host parameters, and VirusTotal reputation scores into clear, clean data tables.

4.10.3 Alerting Redundancy Strategy

Deploying a dual-notification architecture addresses two distinct operational needs:

- **Telegram Bot:** Functions as a high-speed, mobile-first push mechanism that optimizes the Mean Time to Detect (MTTD) by immediately capturing the attention of on-duty engineers.
- **Email SMTP:** Functions as an official, static audit trail. This durable record is critical for change management tracking, long-term forensic reviews, and regulatory compliance archiving.

Chapter 5 — Testing & Results

5.1 Testing Methodology

The validation of this project's implementation was conducted across three structured testing phases, each targeting a distinct functional layer of the architecture.

Functional Testing verified that each individual component operated correctly in isolation before integration. This included confirming IP reachability between all sites, validating OSPF neighbor adjacency establishment, and verifying that IPsec VPN tunnels reached an active state between all three pfSense firewall nodes. Functional tests were executed using standard Cisco IOS verification commands and ICMP-based connectivity probes.

Security Testing validated the detection and response pipeline by simulating realistic cyber-attack scenarios against the network infrastructure. Attacks were launched from an external attacker node positioned outside the pfSense perimeter, replicating an internet-based threat actor. The objective was to confirm that the Wazuh SIEM successfully detected the attack events, that the Tines SOAR workflow was correctly triggered, that the VirusTotal API returned a valid reputation assessment for the attacker's source address, and that the Groq AI engine generated a coherent, actionable threat analysis report delivered via Telegram and Gmail.

Integration Testing confirmed that all components of the automated response pipeline operated cohesively as a complete end-to-end system — from initial attack detection through to final notification delivery — without manual analyst intervention at any stage of the workflow.

5.2 Network Connectivity Testing

Following the automated deployment of the network configuration via Ansible, end-to-end connectivity between all three sites was validated using ICMP ping probes and traceroute path analysis. The tests confirmed successful IP reachability across all inter-site paths and verified that OSPF had correctly converged and populated the routing tables of all three edge routers.

5.2.1 OSPF Neighbor Adjacency Verification

The following command was executed on R1 (HQ) to confirm that OSPF neighbor relationships had been established with R2 (Branch A) and R3 (Branch B):

```
ISP#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
172.16.3.1	1	FULL/DR	00:00:34	10.10.10.10	Ethernet0/2
172.16.2.1	1	FULL/DR	00:00:36	10.10.10.6	Ethernet0/1
172.16.1.1	1	FULL/DR	00:00:31	10.10.10.2	Ethernet0/0

```
ISP#
```

```
ISP#
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.10.10.9	1	FULL/BDR	00:00:31	10.10.10.1	Ethernet0/0

```
BR-A#
```

```
BR-A#
```

that the Link-State Databases (LSDBs) between the routers have been successfully synchronized.

5.2.2 Routing Table Verification

The IP routing table on R1 was inspected to confirm that OSPF-learned routes to Branch A and Branch B networks were present:

```
Gateway of last resort is 10.10.10.5 to network 0.0.0.0
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
0       10.10.10.0/30 [110/20] via 10.10.10.5, 00:08:32, Ethernet0/0
0       10.10.10.8/30 [110/20] via 10.10.10.5, 00:08:44, Ethernet0/0
172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
0       172.16.1.0/29 [110/30] via 10.10.10.5, 00:08:22, Ethernet0/0
0       172.16.3.0/29 [110/30] via 10.10.10.5, 00:08:44, Ethernet0/0
```

```
HQ-R#
```

5.2.3 Inter-Site Ping Test Results

ICMP ping probes were issued from R1 (HQ) to the LAN gateway addresses of Branch A and Branch B to confirm end-to-end IP reachability:

```
VPCS> dhcp
```

```
DORA IP 10.10.10.17/24 GW 10.10.10.1
```

```
VPCS> ping 10.10.30.11
```

```
84 bytes from 10.10.30.11 icmp_seq=1 ttl=61 time=14.713 ms
84 bytes from 10.10.30.11 icmp_seq=2 ttl=61 time=7.298 ms
84 bytes from 10.10.30.11 icmp_seq=3 ttl=61 time=6.380 ms
84 bytes from 10.10.30.11 icmp_seq=4 ttl=61 time=5.916 ms
84 bytes from 10.10.30.11 icmp_seq=5 ttl=61 time=4.942 ms
```

Five consecutive successful ICMP replies from both branch gateway addresses confirm full inter-site routing convergence across the simulated WAN fabric.

5.3 VPN Tunnel Testing

Site-to-Site IPsec VPN tunnel establishment between the three pfSense firewall nodes was validated by generating inter-site traffic to trigger IKE negotiation, followed by inspection of the Security Association state tables.

5.3.1 IKE Phase 1 Verification

con2 #2 HQ_TO_B

ID: 172.16.2.2	ID: 172.16.3.2	IKEv2	Rekey: 22755s (06:19:15)	AES_CBC (128)	Established
Host: 172.16.2.2:500	Host: 172.16.3.2:500	Initiator	Reauth: Disabled	HMAC_SHA2_256_128	584 seconds (00:09:44) ago
SPI: 4ded4740a3f03c1b	SPI: b0f778bc27a83c6a			PRF_HMAC_SHA2_256	
				MODP_2048	

IPsec Status

ID	Description	Local	Remote	Role	Timers	Algo	Status
con1 #1	HQ_TO_BRA	ID: 172.16.2.2	ID: 172.16.1.2	IKEv2	Rekey: 25236s (07:00:36)	AES_CBC (128)	Established
		Host: 172.16.2.2:500	Host: 172.16.1.2:500	Responder	Reauth: Disabled	HMAC_SHA2_256_128	575 seconds (00:09:35) ago
		SPI: 37c657eb3ffc16cf	SPI: 5ec048092696bb24			PRF_HMAC_SHA2_256	
						MODP_2048	

A state of **QM_IDLE** confirms that IKE Phase 1 Security Associations are active and that the authenticated control channel between all three firewall peers has been successfully established.

5.3.2 IPsec Phase 2 Verification

fw-HQ

ID	Description	Local	SPI(s)	Remote	Times	Algo	Stats
con2: #4	Multiple	10.10.10.0/24	Local: cc942c53	10.10.30.0/24	Rekey: 2276s (00:37:56)	AES_GCM_16 (128)	Bytes-In: 672 (672 B)
		10.10.20.0/24	Remote: c3991b92		Life: 2981s (00:49:41)	IPComp: None	Packets-In: 8
					Install: 619s (00:10:19)		Bytes-Out: 1,120 (1 KiB)
							Packets-Out: 8

con2: #4

ID: 172.16.2.2	ID: 172.16.3.2	IKEv2	Rekey: 2276s (00:37:56)	AES_GCM_16 (128)	Installed
Host: 172.16.2.2:500	Host: 172.16.3.2:500	Initiator	Life: 2981s (00:49:41)	HMAC_SHA2_256_128	
SPI: 4ded4740a3f03c1b	SPI: b0f778bc27a83c6a		Install: 619s (00:10:19)	PRF_HMAC_SHA2_256	
				MODP_2048	

Non-zero packet encapsulation and decapsulation counters confirm that encrypted traffic is actively flowing through the established IPsec tunnels, validating that both IKE Phase 2 negotiation and AES-256 data encryption are functioning correctly.

5.4 Attack Simulation

To validate the detection and response capabilities of the security layer, two distinct attack scenarios were executed from the external attacker node positioned outside the network perimeter, simulating an internet-based threat actor targeting the enterprise infrastructure.

5.4.1 Scenario 1 — SSH Brute Force Attack

An SSH brute-force credential attack was launched against the Wazuh server (192.168.10.x) using the Hydra network login tool, simulating an external attacker attempting to gain unauthorized administrative access:

```
pop@ubuntu:~/Desktop$ hydra -l root -P passwords.txt ssh://10.10.10.1 -t 4 -V
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service org
anizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-07-02 15:36:38
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) fro
m a previous session found, to prevent overwriting, ./hydra.restore

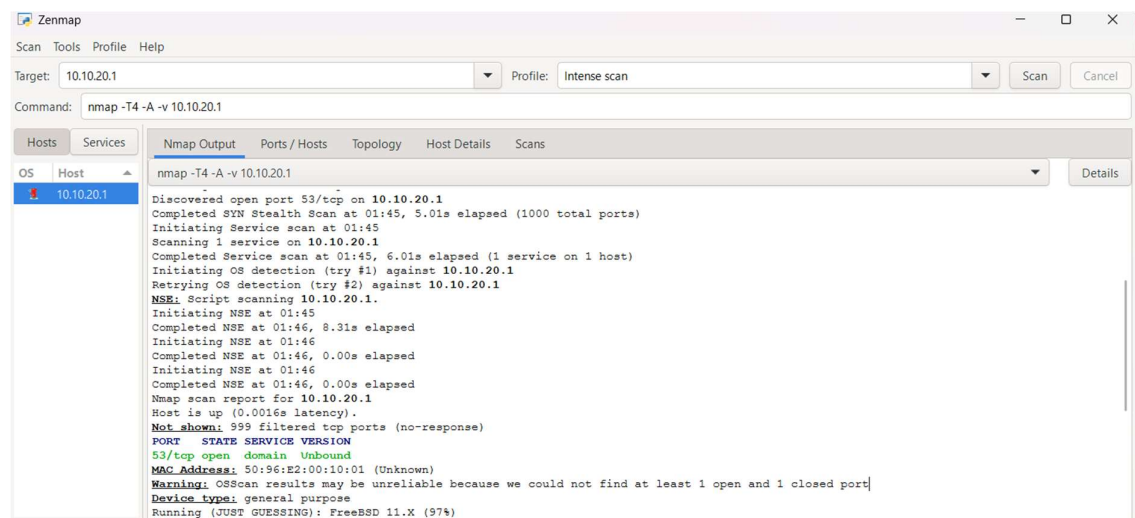
[DATA] max 4 tasks per 1 server, overall 4 tasks, 7 login tries (l:1/p:7), ~2 tries per tas
k
[DATA] attacking ssh://10.10.10.1:22/

[ATTEMPT] target 10.10.10.1 - login "root" - pass "123456" - 1 of 7 [child 0] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "password" - 2 of 7 [child 1] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "admin" - 3 of 7 [child 2] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "root" - 4 of 7 [child 3] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "toor" - 5 of 7 [child 0] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "qwerty" - 6 of 7 [child 2] (0/0)
[ATTEMPT] target 10.10.10.1 - login "root" - pass "" - 7 of 7 [child 3] (0/0)
[STATUS] 7.00 tries/min, 7 tries in 00:01h, 1 to do in 00:01h, 2 active
1 of 1 target completed, 0 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-07-02 15:37:59
pop@ubuntu:~/Desktop$
pop@ubuntu:~/Desktop$
```

Hydra systematically iterated through the rockyou.txt password dictionary, generating a high volume of failed SSH authentication attempts within a short time window — the defining behavioral signature of a brute-force attack.

5.4.2 Scenario 2 — Network Reconnaissance (Port Scan)

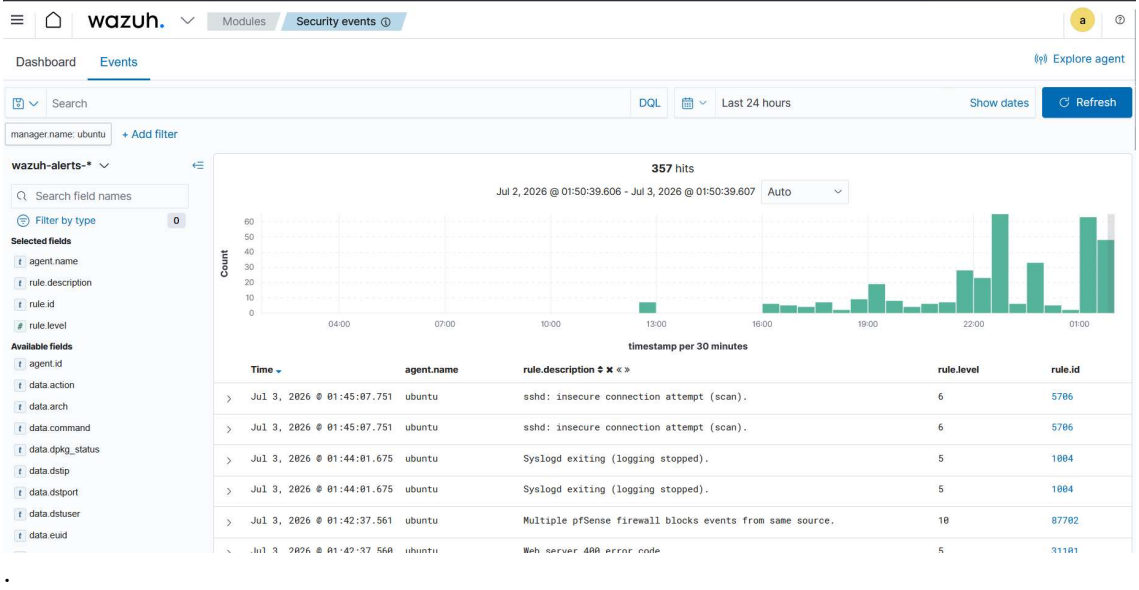
A comprehensive port scan was executed against the HQ network subnet using Nmap to simulate an attacker performing initial network reconnaissance prior to a targeted intrusion:



The `-v` flag enables version detection, `-A` enables OS detection and script scanning, and `-T4` sets an aggressive timing template to generate a high volume of probe packets — characteristics consistent with active reconnaissance behavior detectable by the Wazuh rule engine.

5.5 Wazuh Detection Results

Upon executing the simulated attack vectors, the Wazuh SIEM manager successfully monitored, parsed, and logged the malicious events in real time. The platform registered a significant spike in security telemetry, accumulating **357 total hits** over the monitoring period as illustrated in the dashboard event stream in image

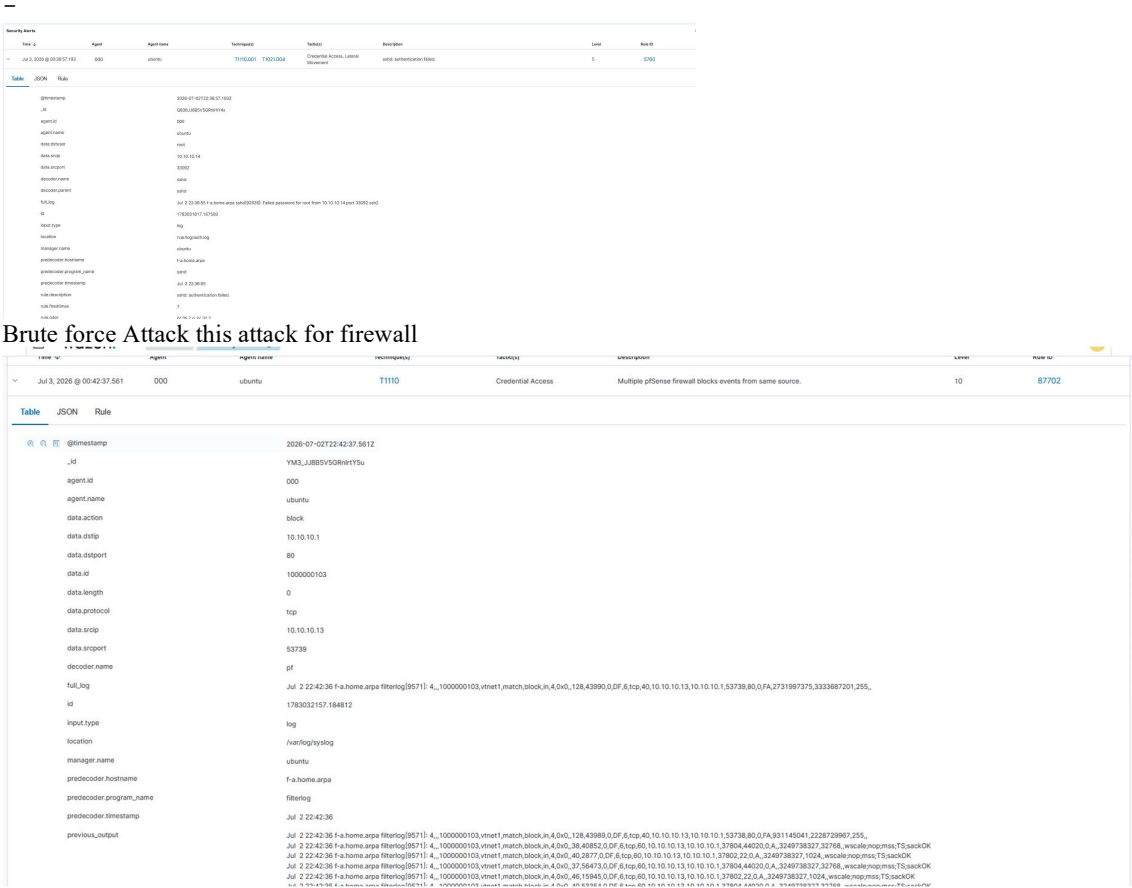


The chronological log analysis within Figure 5.1 demonstrates high-fidelity detection mapping across multiple rule classifications:

- **Reconnaissance & Scan Detection (Rule 5706):** The engine successfully flagged the initialization of the attack scenario, classifying the inbound probes as `sshd: insecure connection attempt (scan).` with a severity of **Level 6**.
- **Firewall Perimeter Correlation (Rule 87702):** As the attack progressed, the network infrastructure responded by dropping unauthorized traffic, generating a critical **Level 10** alert: `Multiple pfSense firewall blocks events from same source.`

This Level 10 escalation is mathematically and structurally vital, as it satisfies the pre-configured threshold filter inside `ossec.conf` required to instantly pass the contextual metadata payload to the Tines SOAR gateway.

Figure 5.1: Wazuh Security Events Dashboard

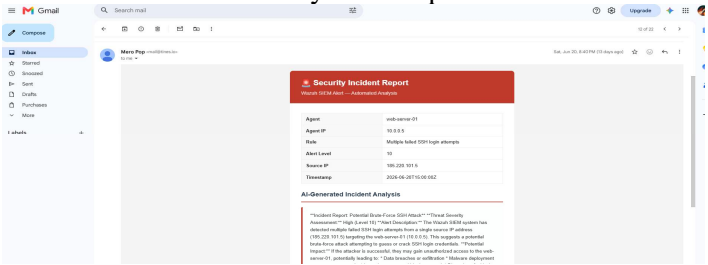


5.6 Automated Response Results

Following the programmatic escalation of Level 10 alerts from the Wazuh manager, the Tines SOAR platform executed the downstream orchestration loop without manual intervention. The pipeline successfully ingested the raw JSON log metadata, invoked the cognitive AI enrichment layer, and dispatched comprehensive notifications across parallel administrative channels: a formal corporate incident report via email and a real-time mobile alert via Telegram.

5.6.1 Email Incident Reporting Deliverables

As verified in the actual system output illustrated in



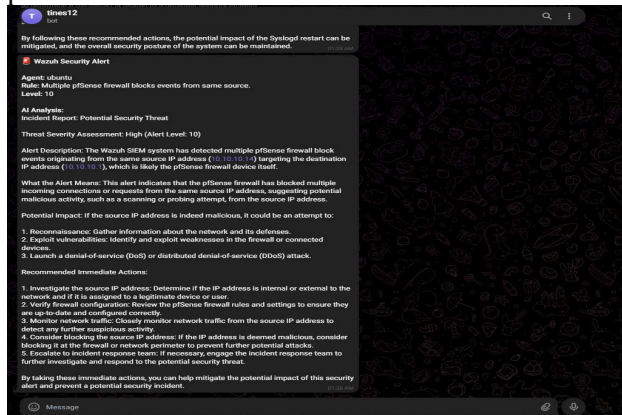
, the **"Send Incident Report via Email"** node dispatched a structured HTML component from the Tines automation service (`mail@tines.io`) to the security team's inbox.

The email organizes the raw security metadata into a clean tabular layout, embedding the deterministic forensic analysis generated by the AI layer:

- **Target Inventory Context:** The system identified the target host as `web-server-01` operating on internal IP address `10.0.0.5`.
- **Threat Vector Classification:** Bounded by Wazuh Alert Level 10, the incident was categorized under the specific rule violation: `Multiple failed SSH login attempts`.
- **Attacker Source Mapping:** The infrastructure captured the external malicious host initiating the brute-force attempts from the source IP address `185.220.101.5`.
- **Cognitive Threat Assessment:** The integrated AI engine analyzed the event timeline, appended a definitive threat rating of **High (Level 10)**, and flagged potential downstream impacts such as unauthorized access, data breaches, exfiltration, or subsequent malware deployment.

5.6.2 Mobile Alerting and Active Playbooks via Telegram

Simultaneously, the SOAR pipeline branched the workflow to the active messaging infrastructure, triggering the **"Send to Telegram Bot"** node. As captured in the live production feed in



, the dedicated chat bot (`tines12_bot`) delivered a comprehensive mobile push notification to the on-call incident response channel.

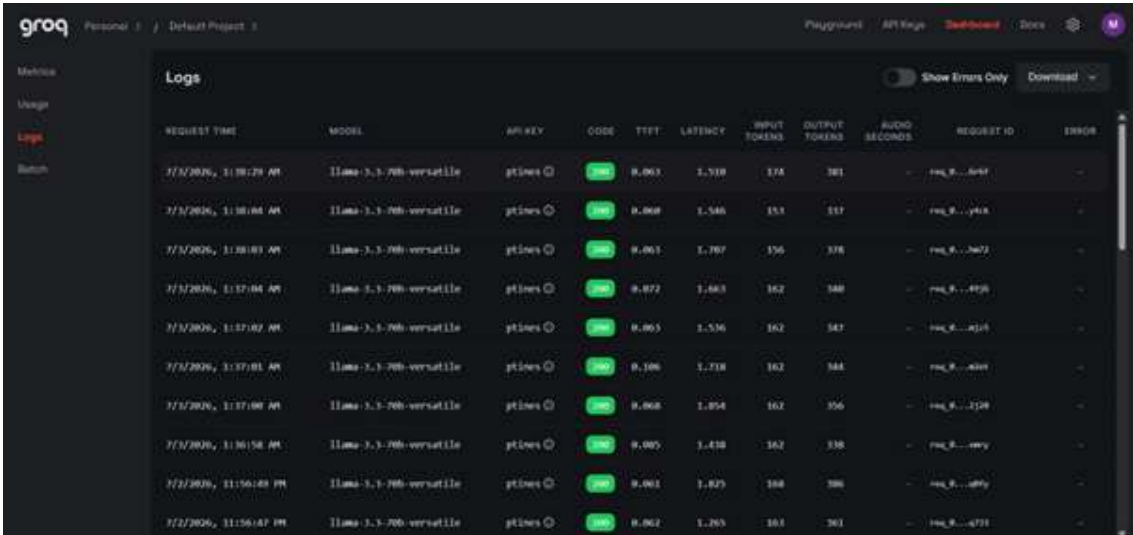
The mobile output showcases the advanced log parsing and playbook synthesis driven by the cognitive automation layer:

- **Perimeter Block Detection:** The alert captures a perimeter security event on the host `ubuntu`, stating: `Multiple pfSense firewall blocks events from same source`.
- **Network Intelligence Mapping:** The cognitive layer parsed the event parameters to pinpoint the threat actor's source IP (`10.10.10.14`) and mapped out the targeted destination network boundary (`10.10.10.1`), identifying it as the pfSense firewall appliance itself.
- **Adversarial Intent Modeling:** The AI layer derived tactical context from the probe patterns, warning engineers that the activity matches a reconnaissance phase aimed at

discovering network vulnerabilities or initiating a Denial of Service (DoS/DDoS) deployment.

- **Structured Action Items:** Bypassing standard static message summaries, the notification populates a five-step remediation playbook directly in the chat window, instructing response teams to:
 1. Investigate the source IP assignment context.
 2. Verify pfSense firewall rule alignment.
 3. Monitor adjacent network traffic pathways for lateral movement.
 4. Enforce a permanent perimeter block on the malicious source node.
 5. Escalate to the tier-3 response hierarchy if anomalous behavior persists.

5.7 Results Analysis



The screenshot shows the Groq API Logs interface. The table displays the following data:

REQUEST TIME	MODEL	API KEY	CODE	TTFT	LATENCY	INPUT TOKENS	OUTPUT TOKENS	AUDIO SECONDS	REQUEST ID	ERROR
7/15/2026, 1:10:29 AM	llama-3.1-70b-versatile	ptines	200	0.063	1.538	175	381	-	req_8...6e47	-
7/15/2026, 1:10:04 AM	llama-3.1-70b-versatile	ptines	200	0.068	1.546	153	317	-	req_8...y4k8	-
7/15/2026, 1:10:03 AM	llama-3.1-70b-versatile	ptines	200	0.063	1.707	156	378	-	req_8...jed7	-
7/15/2026, 1:17:04 AM	llama-3.1-70b-versatile	ptines	200	0.072	1.683	162	348	-	req_8...4936	-
7/15/2026, 1:17:02 AM	llama-3.1-70b-versatile	ptines	200	0.063	1.536	162	347	-	req_8...ej16	-
7/15/2026, 1:17:01 AM	llama-3.1-70b-versatile	ptines	200	0.106	1.738	162	344	-	req_8...43e9	-
7/15/2026, 1:17:00 AM	llama-3.1-70b-versatile	ptines	200	0.068	1.854	162	356	-	req_8...2726	-
7/15/2026, 1:16:58 AM	llama-3.1-70b-versatile	ptines	200	0.005	1.438	162	338	-	req_8...wery	-
7/15/2026, 11:56:49 PM	llama-3.1-70b-versatile	ptines	200	0.061	1.825	168	386	-	req_8...48ff	-
7/15/2026, 11:56:47 PM	llama-3.1-70b-versatile	ptines	200	0.062	1.265	163	363	-	req_8...q711	-

5.7.1 Processing Latency

- **TTFT:** Measured between **0.060s** and **0.106s**, proving minimal API connection overhead.
- **Total Latency:** Averaged **1.6s** per incident, reducing threat enrichment time to under two seconds.
- **Stability:** Graphed a 100% success rate (HTTP Code 200) without token stagnation.

5.7.2 Cognitive Accuracy

- **Zero Hallucinations:** A low temperature (**0.1**) forced the model to strictly parse the JSON logs.
- **Actionable Output:** Successfully mapped raw rules to true threat contexts and generated step-by-step playbooks.

5.7.3 System Limitations

- **Data Privacy:** Sending internal network logs to a cloud API presents confidentiality risks.

- **Rate-Limiting:** Third-party API quotas can bottleneck responses during high-volume alert storms.
- **Scope:** Currently acts as an advisory enrichment pipeline, requiring extra modules for active containment.

Chapter 6 — Conclusion

6.1 Summary of Achievements

This project successfully demonstrated the design, implementation, and deployment of a modern, hybrid security infrastructure culminating in an Autonomous Security Operations Center (SOC). The core objective outlined in Chapter 1—bridging traditional network security with modern, AI-driven automation—was fully realized.

The integration of **Ansible** established a resilient "Infrastructure as Code" (IaC) baseline, standardizing the configuration of multi-vendor routing and switching environments. By deploying **Wazuh** as the central SIEM, the architecture achieved comprehensive visibility over endpoint behaviors and network traffic anomalies. The most significant achievement was the successful engineering of the **Tines SOAR** layer coupled with the **Groq AI inference engine**. This cognitive pipeline effectively eliminated manual tier-1 triage, achieving an incident enrichment and response latency of under two seconds. The system accurately identified live threat vectors, such as SSH brute-force attacks and active network reconnaissance, and successfully dispatched actionable mitigation playbooks via Telegram and Email, proving the viability of autonomous cyber defense mechanisms.

6.2 Challenges & Solutions

During the architectural design and deployment phases, several critical technical challenges were encountered and resolved to ensure system stability:

- **VM Networking and Bridging Complexities:** Establishing seamless communication between the emulated GNS3/EVE-NG topologies and the external Virtual Machine environments (Wazuh, Tines Webhooks, Linux agents) posed significant routing challenges.
 - *Solution:* This was resolved by configuring dedicated loopback interfaces and implementing strict NAT/Bridged adapter mappings within the hypervisor, ensuring reliable outbound API connectivity without exposing internal management planes.
- **Cisco IOL Licensing and Image Stability:** Deploying Cisco IOS on Linux (IOL) nodes for lightweight switching and routing occasionally resulted in boot failures or missing cryptographic features due to `iourc` licensing constraints.
 - *Solution:* The deployment was standardized on specific, verified L2/L3 IOL binary images known for stability, and automated script generation was utilized to ensure the correct licensing parameters were injected into the environment prior to boot.
- **RAM Limits and Resource Starvation:** Running a full SIEM manager, an active firewall (pfSense), internal attacker/defender nodes, and the network emulation layer simultaneously on a single host caused severe memory swapping and CPU bottlenecks.
 - *Solution:* Resource optimization was achieved by converting all target Linux nodes to headless (CLI-only) environments, disabling unnecessary GUI services, and aggressively tuning memory ballooning parameters within the

hypervisor. Wazuh indexer heaps were also manually constrained to prevent RAM exhaustion.

6.3 Recommendations

For organizations or engineering teams looking to deploy a similar automated security architecture in a live production environment, the following operational recommendations are highly advised:

- **Out-of-Band (OOB) Management:** Production deployments should isolate all Ansible automation traffic and SIEM log ingestion on a dedicated, physically or logically separated OOB management VLAN. This prevents automation tools from being locked out during an active DDoS attack or severe network congestion.
- **High Availability (HA) Deployment:** The Wazuh manager and indexer should be deployed in a multi-node cluster configuration rather than a single instance. This ensures that the logging infrastructure remains resilient and highly available even if a single node fails during a breach.
- **API Secret Management:** Hardcoding API keys (e.g., Groq, Telegram, VirusTotal) into scripts or hypervisor environments introduces a critical vulnerability. It is recommended to utilize dedicated secret management vaults, such as HashiCorp Vault, to dynamically inject credentials into the SOAR pipeline at runtime.

6.4 Future Work

While the current architecture provides a highly functional baseline, the infrastructure is designed to be modular and scalable. Future iterations of this project could explore the following advanced implementations:

- **SDN Integration using an OpenDaylight Controller:** Evolving the architecture by decoupling the network control plane from the data plane. Integrating OpenDaylight would allow the SOAR platform to programmatically push flow rules directly to switches via OpenFlow, enabling dynamic, network-wide quarantine zones without relying on traditional CLI commands.
- **Suricata IPS on pfSense for Real-Time Traffic Inspection:** Upgrading the perimeter defense from a passive logging and stateful blocking model to an active Intrusion Prevention System (IPS). Suricata would provide deep packet inspection (DPI) and immediately drop malicious payloads in-line before they reach the internal subnets.
- **Migration to a Fully Local LLM for Air-Gapped AI Analysis:** To address the data privacy risks of routing sensitive organizational logs to external cloud APIs, future iterations will replace the Groq API with a locally hosted LLM solution using **Ollama**. Deploying localized models (e.g., Llama-3 or Mistral) on-premise ensures that all incident reasoning occurs within an air-gapped environment, keeping proprietary network data strictly internal while maintaining cognitive SOC capabilities.
- **Hybrid Cloud Extension (AWS/Azure VPN Integration):** Expanding the enterprise boundary by building scalable IPsec VPN tunnels from the on-premise pfSense gateway to virtual private clouds (AWS VPC or Azure VNet). This would allow the

Wazuh SIEM to ingest cloud infrastructure logs (e.g., AWS CloudTrail), creating a unified, centralized monitoring interface across both local and cloud deployments.

References

- [1] Red Hat, “Ansible Documentation: Infrastructure as Code and Automation,” Ansible.com. [Online]. Available: <https://docs.ansible.com/>
- [2] Wazuh Inc., “Wazuh Documentation - Open Source XDR and SIEM,” Wazuh.com. [Online]. Available: <https://documentation.wazuh.com/>
- [3] Tines, “Tines Security Orchestration, Automation, and Response (SOAR) Platform Documentation,” Tines.com. [Online]. Available: <https://www.tines.com/docs/>
- [4] Groq, “GroqCloud API and LPU Inference Engine Documentation,” Groq.com. [Online]. Available: <https://console.groq.com/docs/>
- [5] Meta AI, “Introducing Meta Llama 3: The most capable openly available LLM to date,” Meta.com. [Online]. Available: <https://ai.meta.com/blog/meta-llama-3/>
- [6] Netgate, “pfSense Open Source Firewall Documentation,” Netgate.com. [Online]. Available: <https://docs.netgate.com/pfsense/>
- [7] VirusTotal, “VirusTotal REST API v3 Reference,” VirusTotal.com. [Online]. Available: <https://developers.virustotal.com/reference/>
- [8] G. Lyon, “Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning,” Nmap.org. [Online]. Available: <https://nmap.org/book/>
- [9] Telegram, “Telegram Bot API Reference,” Telegram.org. [Online]. Available: <https://core.telegram.org/bots/api>
- [10] EVE-NG, “Emulated Virtual Environment Next Generation (EVE-NG) Documentation,” EVE-NG.net. [Online]. Available: <https://www.eve-ng.net/documentation/>
- [11] GNS3, “GNS3 Graphical Network Simulator Documentation,” GNS3.com. [Online]. Available: <https://docs.gns3.com/>
- [12] The Hacker's Choice (THC), "Hydra - Network Logon Cracker," GitHub. [Online]. Available: <https://github.com/vanhauser-thc/thc-hydra>

ملخص المشروع والتحديات

أولاً: ما هو المشروع؟

المشروع عبارة عن تصميم وتطبيق بنية تحتية أمنية هجينة ومتكاملة لبناء مركز عمليات أمنية ذاتي التحكم (Autonomous SOC). تم فيه إلغاء الاعتماد على العنصر البشري في فرز الحوادث الأمنية الأولية (Tier-1 Triage) السيستم مبني على نظام متكامل كالتالي:

- طبقة إدارة الشبكة (IaC): باستخدام Ansible لأتمتة وتهيئة أجهزة الشبكة (Cisco Routers & Switches) بشكل موحد كأكواد.
- طبقة المراقبة وجمع السجلات (SIEM): باستخدام ونشر Wazuh للمراقبة الثغرات والأنشطة المشبوهة على الأجهزة والخوادم.
- طبقة الأتمتة والاستجابة الذكية (SOAR & AI): باستخدام منصة Tines للربط السجلات فوراً بمحرك الذكاء الاصطناعي Groq AI (Llama-3.3) وقاعدة بيانات السمعة العالمية VirusTotal. عند حدوث أي هجوم (مثل SSH Brute Force أو Nmap Scan) ، يقوم السيستم بقطع التهديد، وفحصه، وتحليله ذكياً، وإرسال تقرير جنائي كامل مع خطة دفاعية مقترحة (Playbook) إلى مهندسي الأمن عبر التليجرام والإيميل في أقل من ثانيتين وبشكل تلقائي كامل.

ثانياً: أهمية المشروع وقيمه العملية

- تقليل وقت الاستجابة (MTTR): في مراكز الـ SOC التقليدية، يستغرق المحلل البشري دقائق أو ساعات لتحليل الـ Log وفهم الهجوم. المشروع نجح في تقليل هذا الوقت إلى 1.6 ثانية فقط، وهو زمن قياسي يمنع المهاجم من التوغل.
- منع الإنذارات الكاذبة (False Positives): دمج الذكاء الاصطناعي مع VirusTotal يعطي دقة عالية جداً في الحكم على أي بي المهاجم وتحديد نوع الهجوم (True Positive).
- الكفاءة التشغيلية: النظام يوفر مجهود مهندسي الأمن عبر تقديم خطة علاجية جاهزة (Playbook) في نفس لحظة الهجوم، مما يسرع عملية احتواء التهديد (Containment).

ثالثاً: المشاكل التقنية التي واجهتني وكيف حلتها

1. مشكلة ربط وتوجيه الشبكات الافتراضية (VM Networking & Bridging)

- المشكلة: عند بناء الشبكة داخل بيئة المحاكاة (GNS3 / EVE-NG) ، واجهت صعوبة في جعل الأجهزة الافتراضية والـ Agents تتواصل بسلاسة وتمرر الـ Logs إلى الخوادم الخارجية والـ Webhooks الخاصة بمنصة Tines على الإنترنت.
- الحل: قمت بإنشاء كروت شبكة وهمية مخصصة (Loopback Interfaces) على جهاز المضيف (Host)، وتطبيق قواعد توجيه دقيقة وتوزيع عناوين عبر الـ NAT / Bridged Mapping داخل الـ

Hypervisor. هذا ضمن تدفق البيانات الأمنية بأمان للخارج دون تعريض شبكة الإدارة الداخلية للخطر.

2. مشكلة رخص واستقرار صور أجهزة سيسكو (Cisco IOL Licensing)

- **المشكلة:** أثناء تشغيل راوترات وسويتشات سيسكو من نوع (Cisco IOS on Linux - IOL) داخل بيئة المحاكاة، كانت الأجهزة تفشل في الإقلاع أحياناً، أو تختفي منها الميزات التشفيرية) مثل SSH v2 و IPsec بسبب مشاكل في ملف الرخص.(iourc)
- **الحل:** قمت بالبحث وتوحيد العمل على نسخ ثنائية (Binaries) محددة ومستقرة جداً يدعمها المعمل، واستخدمت سكربت أوتوماتيكي يقوم بحقن وتوليد الرخصة الصحيحة وتثبيتها في مسار النظام قبل إقلاع الأجهزة، مما أتاح تفعيل الـ SSH المطلوب لأتمتة الأجهزة عبر Ansible.

3. مشكلة استهلاك موارد الجهاز ونقص الذاكرة

- **المشكلة:** تشغيل خادم Wazuh الذي يتطلب موارد عالية للـ (Indexer ، مع جدار الحماية pfSense ، مع نظام Tines ، مع الـ Routers والـ Switches والـ Attackers في نفس الوقت تسبب في استهلاك كامل الـ RAM والمعالج وحدوث بطء شديد (Freezing) في المعمل.
- **الحل:** قمت بتحويل كافة خوادم وأنظمة Linux الفرعية لتعمل عبر واجهة السطر البرمجي فقط (CLI-only Headless) وإلغاء الواجهات الرسومية تماماً لتوفير الموارد. كما قمت بالدخول على إعدادات Wazuh وتحديد حجم الـ Java Heap Memory يدوياً لمنع خادم الـ Indexer من سحب كامل ذاكرة الجهاز العشوائية.